VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Riadenie mobilnej satelitnej antény

Control Of Portable Satellite Antenna

2015 Bc. Ľubomír Wenzl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Ľubomír Wenzl**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Řízení mobilní satelitní antény
Control Of Portable Satellite Antenna

Zásady pro vypracování:

Cílem práce je vytvořit program pro počítač Raspberry-PI, který bude prostřednictvím svých periférií ovládat mobilní satelitní anténu. Hlavní úkol program je nastavit elevaci a azimut satelitní atnény, tak aby bylo možné přijímat satelit Astra na pozici 23.5 východně.

1. Popište dodaný hardware, stručně zdokumentujte primcip komunikace s periferiemi.
2. Vytvořte program pro Raspberry-PI, který otestuje přítomnost a správnou funkci všech periférií.
3. Vytvořte program, který bude používat rozhraní ncurses a bude jím možné ručně ovládat azimut a elevaci antény, přičemž budou zobrazována data z čidla náklonu, čidla relativní a absolutní pozice antény, kompasu a úroveň signálu na anténě.
4. Vytvořte program, který automaticky nastaví pozici antény, tak aby byl co nejlépe zajištěn příjem z družice Astra na pozici 23.5 východně.

Seznam doporučené odborné literatury:

[1] SKOČOVSKÝ, Luděk a Scott JERNIGAN. Linux: dokumentační projekt. 4., aktualiz. vyd. Překlad Ivo Fořt, David Krásenský. Brno: Computer Press, 2007, 1334 s. ISBN 978-80-251-1525-1.
[2] UPTON, Eben a Gareth HALFACREE. Raspberry Pi user guide. 2. aktualiz. vyd. Chichester, England: John Wiley, c2012. xiv, 248 p. ISBN 11-184-6446-X.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Súhlasím so zverejnením tejto diplomovej práce podľa požiadavkov čl. 26, odst. 9 *Študijnej a skúšobnej rady pre štúdium v inžinierskych programoch VŠB-TU Ostrava.*

V Bratislave 16. Apríla 2015 .....

Dolu podpísaný Bc. Ľubomír Wenzl čestne vyhlasujem, že som diplomovú prácu Riadenie mobilnej satelitnej antény vypracoval sám, pod vedením Ing. Dávida Seidla, Ph.D. Prácu som vypracoval na základe poznatkov získaných počas štúdia a použitej literatúry uvedenej na konci tejto práce v časti Literatúra.

V Bratislave 16. Apríla 2015 .....

Touto cestou by som rád poďakoval môjmu vedúcemu pánovi Ing. Dávidovi Seidlovi, Ph.D. za jeho vedenie a odborné rady, ktoré mi pomáhali pri tvorbe tejto diplomovej práce. Moja vďačnosť patrí aj mojej rodine a mojím priateľom, ktorí mi s prácou pomohli, pretože bez nich by táto práca nevznikla.

Abstrakt

Práca prináša a systematizuje pohľad na geostacionárne satelity na základe ktorých sa skonštruuje pozemná prijímajúca anténa v mobilnom prevedení. Hlavným cieľom tejto diplomovej práce bude preto popis hardvérového vybavenia mobilnej satelitnej antény a analýza komunikácie medzi jednotlivými rozhraniami použitých senzorov, ktoré spracováva ovládací kontrolér. Na základe týchto dát je nastavená pozícia paraboly mobilnej satelitnej antény s cieľom zamerať najlepší signál družice Astra na pozícii 23,5 východne. V tejto práci sa sústredím na použité implementačné postupy pri tvorbe utility, ktorá je nasadená v ovládacom kontroléri, ktorý predstavuje počítač Raspberry Pi.

Kľúčové slová: senzor, Astra, signál, Raspberry Pi, satelit

Abstract

The work presents a systematized view of geostationary satellites on the basis of which it is constructed ground antenna receiver in a mobile format. The main objective of this thesis will be a description of hardware equipment of mobile satellite antenna and analysis of communication between the different interfaces of used sensors which are controlled by processing controller. On the basis of these data is set position of the mobile satellite dish antenna in order to focus the best signal on satellite Astra 23.5 East. In this work i will focus on the implementation procedures for creation utility, which is deployed in the controller driver that represents the computer Raspberry Pi.

Keywords: sensor, Astra, signal, Raspberry Pi, satellite

Zoznam použitých skratiek a symbolov

ARM	–	Acorn RISC Machine
I2C	–	Internal-Integrated Circuit
SPI	–	Serial Peripheral Interface
SDA	–	Serial Data Line
SCL	–	Serial Clock Line
ACK	–	Acknowledge
SAT	–	Satelit
CLI	–	Command Line Interface
GUI	–	Graphics User Interface
GPS	–	Global Positioning System
SoC	–	System On Chip
RPi	–	Raspberry Pi
GPIO	–	General Purpose Input/Output (v prekl. univerzálny vstup / výstup)
GEO	–	Geostationary Earth Orbit (v prekl. geostacionárna zemská orbita)
LNB	–	Low Noise Block (v prekl. nízko šumový konvertor)
DVB-S	–	Digital Video Broadcast Satellite
WSN	–	Wireless Sensor Network (v prekl. bezdrôtová senzorová sieť)

Obsah

1 Úvod **5**

2 Teória geostacionárnych satelitov **7**
2.1 Charakteristika 7
2.2 Distribúcia TV signálu 8
2.3 LNB konvertor 10

3 Konštrukcia mobilnej satelitnej antény **11**
3.1 Potrebné senzory 11
3.2 Použité senzory 12
3.3 Ovládací kontrolér 16
3.4 Mechanická konštrukcia 19

4 Komunikácia periférii **23**
4.1 Zbernica I2C 23
4.2 GPS štandard NMEA 0183 26

5 Návrh softvérového modelu **29**
5.1 Špecifikácia požiadaviek 29
5.2 Prípady použitia aplikácie 29
5.3 Sekvenčný diagram východzej pozície 30
5.4 Diagram aktivít pri použití DVB-S 32

6 Implementácia **34**
6.1 Konfigurácia Raspbian OS 34
6.2 Implementačný jazyk a použité nástroje 35
6.3 Použité prídavné knižnice 37
6.4 Kompilácia a krížená kompilácia 42
6.5 Spôsoby implementácie niektorých funkcií 43
6.6 Ladenie a testovanie 45
6.7 Ďalšia práca 47

7 Záver **48**

8 Literatúra **49**

Prílohy **51**

A Elektronická príloha **52**

Zoznam tabuliek

1 Vysvetlivky k obrázku natáčania paraboly 21
2 Mapovanie I2C adries 24
3 Rozdiel medzi označením pinov v aplikácii *Gpio* a knižnicou BCM 2835 . 35
4 Mapovanie GPIO pinov 41

Zoznam obrázkov

1 GEO vzdialenosť k pomeru rádiusu Zeme (zdroj fi.muni.cz) 7
2 Pokrytie zemského povrchu družicou Astra 3B (zdroj satbeams.com) . . . 9
3 Bloková schéma zapojenia komponentov (zdroj vlastný) 11
4 Diagram rozmiestnenia komponentov na RPi v.1 (zdroj elinux.org) 17
5 Označenie pinov GPIO na RPi v.1 (zdroj advamation.com) 18
6 Natáčanie paraboly (zdroj vlastný) 21
7 Možné blokové zapojenia I2C čipov (zdroj fl.hw.cz) 24
8 Kompletný I2C dátový prenos (zdroj nxp.com) 26
9 Use Case diagram vyvíjaného programu (zdroj vlastný) 30
10 Sekvenčný diagram nastavenia elevácie (zdroj vlastný) 31
11 Sekvenčný diagram nastavenia relatívneho azimutu (zdroj vlastný) 32
12 Aktivity diagram použitia DVB-S karty (zdroj vlastný) 33
13 Grafické rozhranie SatBerry (zdroj vlastný) 46

Zoznam výpisov zdrojového kódu

1 Funkcia na pozastavenie behu vlákna 40
2 Smerovacia funkcia azimutu podľa smeru a počtu stupňov 43

1 Úvod

V dnešnom konzumnom spôsobe života chcú ľudia pristupovať k svojim obľúbeným televíznym programom kdekoľvek a kedykoľvek a aj počas pohybu. Tak, ako využívajú mobilitu pri smartfónoch, tak by chceli využívať mobilitu pri sledovaní televízie. To im môže ponúknuť mobilná satelitná anténa, ktorá sa automaticky natáča k zvolenej družici umiestnenej na geostacionárnej orbitálnej výške a s cieľom, naladiť čo najlepší signál. Takáto anténa sa umiestni na strechu vozidla a umožňuje tak zachytávať živé vysielanie. Nemusí ísť len o vozidlá spôsobilé na premávku po pozemných komunikáciách, ale aj o jachty, lode, alebo vlaky, či akákoľvek verejná doprava okrem lietadiel.

Zvyšuje sa úroveň spokojnosti zákazníkov a užívateľský komfort. Niektoré satelitné antény dokážu nielen prijímať, ale aj vysielať, čo na nich umožňuje prevádzkovať širokú škálu komerčných aplikácií vrátané internetu.

Väčšinu mobilných satelitných antén však tvoria rozmerné a pri prenášaní nepraktické paraboly. Preto výrobcovia antén vymysleli ich zmenšené prevedenie, ktoré nezaberá toľko priestoru a je samostatné, čo sa týka jeho naladenia. Tieto druhy antén sa vedia prispôsobovať svojej polohe. Majú k dispozícii polohové senzory, aby si vedeli určiť svoju zemepisnú šírku, dĺžku, natočený smer vzhľadom na odchýlenie od severu a nakoniec uhol záklonu.

K hardvérovému vybaveniu, okrem pripojených snímačov, neodmysliteľne patrí kontrolér, ktorý to celé riadi. V tejto implementácií je použitý britský miniaturizovaný počítač veľkosti zhruba platobnej karty, nazývaný Raspberry Pi, ktorý je vo svete techniky známy, ako najlacnejší a viacúčelový počítač. Navrhli ho na univerzite v Cambridgi s cieľom podporiť výuku na školách, kde si miestni učitelia všimli, že s postupným príchodom nových generácií počítačov a mobilných zariadení, nieje potrebné toľko programovať a zhoršuje sa tak kvalita programátorských schopností žiakov. Projekt sa veľmi rozšíril a má široké uplatnenie, čo dokazuje aj neustále pribúdanie implementačných projektov na internete. Jedným z webových serverov, čo ich združuje je napríklad [1].

Okrem použitého hardvéru bude rozoberaná aj komunikácia medzi rôznymi periférami. Bude popísaný protokol I2C, používaný na obojsmernej sériovej zbernici o dvoch vodičoch, prostredníctvom ktorého komunikujú rôzne druhy elektronických zariadení, ako napríklad v tomto prípade kontrolér a snímač záklonu. Ďalším používaným protokolom je NMEA, ktorý určuje štandard pri posielaní dát o polohe pre GPS zariadenia a tieto informácie zoskupuje do viet. Dôležitým rozhraním kontroléru je GPIO, ktoré je používané na ovládanie elektrických motorčekov pri smerovaní paraboly, a preto budú popísané jednotlivé jeho piny.

Vzhľadom na uvedené skutočnosti a fakty budem môcť navrhnúť satelitnú aplikáciu, ktorá bude implementovať navádzacie algoritmy a požadovanú funkcionalitu, ktorú si znázorním pomocou UML diagramov, kde priblížim užívateľovi, ako by mal program vyzerať a čo dokáže.

Po návrhu softvérového modelu budem mať predstavu, aké ciele sa budem snažiť dosiahnuť a s pomocou akých technológií. Medzi tieto technológie patrí zvolený progra-

[1] `http://www.instructables.com/howto/Raspberry+Pi/`

movací jazyk a k nemu prídavné knižnice. Každý z týchto nástrojov popíšem a ukážem na vzorových príkladoch ich prípadné použitie.

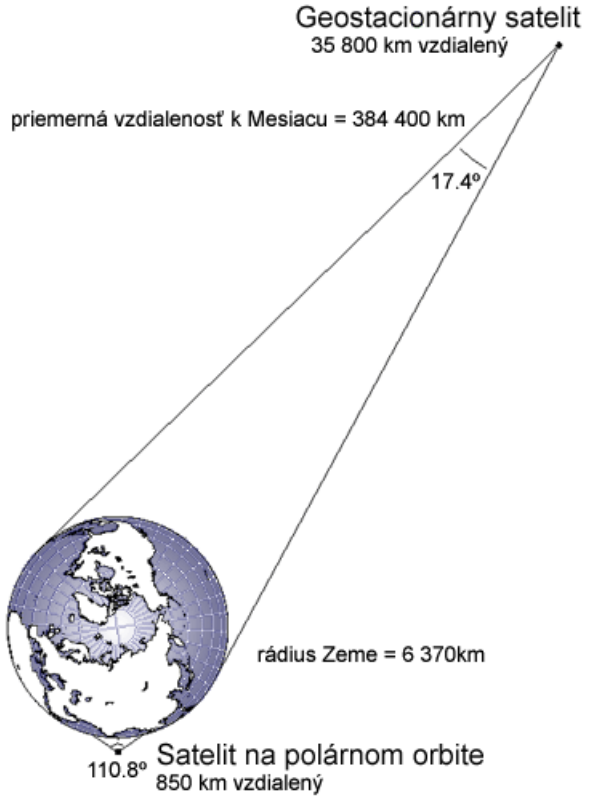
2 Teória geostacionárnych satelitov

Satelity na prenos televízneho signálu sú vynášané na geostacionárnu zemskú dráhu (v skratke GEO), ktorá sa nachádza v orbitálnej výške cca. 36 000 km nad povrchom Zeme a sú označené ako geostacionárne (GEO satelity). Viac o týchto satelitoch a ich distribúcií signálu bude vysvetlené v nasledujúcich podkapitolách.

2.1 Charakteristika

Geostacionárne satelity sú charakterizované tým, že obehnú Zem za rovnakú dobu, za akú sa Zem otočí. To znamená, že satelit z pohľadu zo Zeme sa javí vždy ako stacionárny objekt. Vďaka tomu sa nemusia riešiť zložité sústavy vzájomne sa križujúcich satelitných dráh a synchronizácia prechodov satelitov z pásma do pásma.

Z tak vysokej orbitálnej výšky, ako je 35 800 km, sa Zem tvári ako plochý 2-D „disk“ na úrovni rovníka pod uhlom $17,4^o$, ako znázorňuje nasledujúci obrázok 1.



Obr. 1: GEO vzdialenosť k pomeru rádiusu Zeme (zdroj fi.muni.cz)

Medzi výhody GEO satelitov patria:

- **pevné upevnenie pozemných antén** - pozemná stanica sleduje satelit, ktorý sa pohybuje rovnakou uhlovou rýchlosťou ako Zem.

- **väčšie pokrytie planéty jedným satelitom** - vo výške 35 786 km nad Zemou môže satelit komunikovať približne so štvrtinou Zeme. Takto je možné troma satelitmi pod uhlom 120^o pokryť väčšinu obývaných častí Zeme.

Z toho vyplýva aj niekoľko nevýhod:

- **oblasti blízko pólov sú nepokryté** - z dôvodu pevnej polohy nad rovníkom.
- **potreba vyššej vysielacej energie** - pri výške 35 000 km môže byť signál slabý.
- **oneskorenie prenosu signálu** - ako uvádza zdroj [1], „doba prenosu je značná dokonca aj pri rýchlosti približne 300 000 km/s. Komunikácia medzi dvoma miestami na Zemi priamo pod satelitom, je v skutočnosti $(2x35786)/300000 \approx 0,24s$. Pre ostatné oblasti nie priamo pod satelitom sa táto doba zvyšuje.“

Na GEO sú umiestnené napríklad družice na prenos televízneho, rádiového, ale aj internetového (VSAT - Very Small Aperture Terminals) signálu. Vzhľadom na to, že pokrývajú obrovské územia svojim signálom s pridelenou frekvenciou, tak nie je možné jej opätovné využitie v danej oblasti. To však nevadí pri vše-smernom (angl. broadcast) televíznom vysielaní, keď prijímané stanice si vyžadujú rovnakú kolekciu televíznych programov.

Zdroj:[1]

2.2 Distribúcia TV signálu

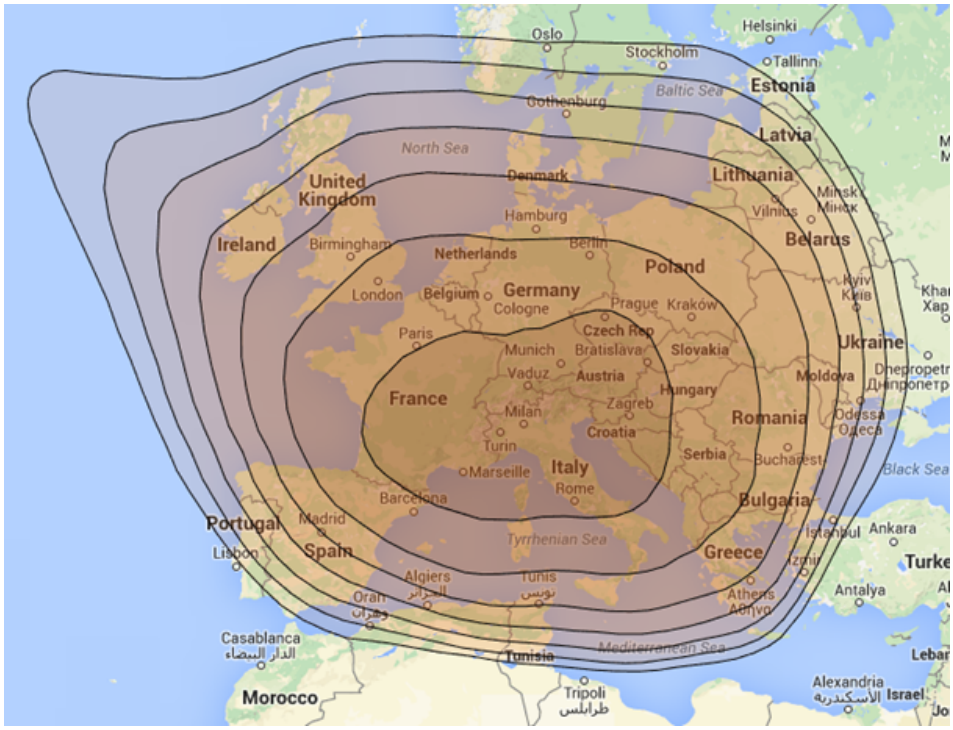
Zo spomenutých výhod v predchádzajúcej podkapitole 2.1 je najdôležitejšia pevné upevnenie pozemných antén, a to pre distribúciu televízneho signálu. Je to z toho dôvodu, aby užívateľ nemusel neustále natáčať svoju parabolu na určenú družicu, ktorá sa pohybuje na obežnej dráhe.

Satelity na obežnej dráhe prijímajú signál z obrovských pozemných parabolických antén, ktoré majú priemer okolo 13 metrov a vysoký zisk a to z toho dôvodu, aby sa znížili straty, ktoré môžu nastať počas prenosu v zhoršených klimatických podmienkach. Tento smer prenosu zo Zeme k satelitu je označovaný ako **uplink**, alebo aj ako Teleport.

Následne sa spracuje signál satelitom, respektíve transpondérom a odošle sa prijímajúcim anténam na Zemi. Tento smer prenosu z družice na Zem je označovaný ako **downlink**.

Zdroj: [2]

Vyznačenie oblasti pokrytia signálom downlink na zemskom povrchu sa nazýva **footprint** (v prekl. stopa) a je znázorňovaná pomocou nepravidelných kriviek podobných vrstevniciam. Ako uvádza Dávidik (2007), „Územie, ktoré je ohraničené takouto „vrstevnicou“ predstavuje priestor, v ktorom je parabola s určitou veľkosťou schopná prijímať kvalitný signál zo satelitu. Posunom k ďalšej „vrstevnici“ na vyžarovacej mapke sa logicky zväčšuje aj veľkosť potrebnej paraboly.“ Príklad satelitnej stopy družice Astra 3B je možné vidieť na nasledujúcom obrázku 2.



Obr. 2: Pokrytie zemského povrchu družicou Astra 3B (zdroj satbeams.com)

Ide o stopu **družice Astra 3B** na pozícii $23,5^o$ východne, ktorá sa rozprestiera nad Európou. Bola vypustená na obežnú dráhu spoločnosťou SES S.A z vesmírneho centra Guiana dňa 21.5.2010. Išlo o model Eurostar-3000, ktorý vyrobila firma EADS Astrium a jej životnosť je odhadovaná na 15 rokov. Na orbite sa drží na úrovni GEO a jej trigonometrický signál je 11702,5 V.

Zdroj: [3]

Pri smerovaní pozemnej paraboly treba dbať na nasledovné faktory:

- **elevácia** - označuje vertikálny výškový uhol parabolickej antény v stupňoch pri pohľade z horizontu.
- **azimut** - označuje horizontálne orientovaný zemepisný uhol v stupňoch medzi určitým smerom a severným smerom, ktorý tvorí nulový stupeň. Inak povedané, jedná sa o uhol pravého, alebo ľavého natočenia antény.
- **magnetická deklinácia** - (odklon) skutočného severného pólu Zeme od magnetického severného pólu. Musí sa zohľadniť pri výpočte azimutu a elevácie prijímacej antény. Mení sa s časom.

Výpočet správneho nastavenia azimutu a elevácie prijímacej antény prebieha v šiestich krokoch. Ako uvádza zdroj [5], podľa ktorého sa dá prakticky a zjednodušene vypočítať hodnoty azimutu a elevácie pozemného satelitu podľa jeho aktuálnej polohy vzhľa-

dom na pozíciu družice, na ktorú sa smeruje. Vychádza z problému dvoch telies a kompaktného algoritmu pre výpočet uhlov. Existujú však aj pre to rôzne webové kalkulačky, napríklad jedna je dostupná na adrese [2].

2.3 LNB konvertor

Pri distribúcií DVB-S signálu, resp. pre príjem z družice k satelitnej pozemnej anténe, je dôležitým prvkom satelitnej antény LNB konvertor. Je to zariadenie, ktoré prijímaný signál z družice zosilnie a prevedie vstupné frekvencie na nižšie hodnoty[3], ktoré už satelitný prijímač pretransformuje do audiovizuálnej podoby.

LNB konvertory sa rozlišujú podľa:

- počtu výstupov a zapojených satelitných prijímačov:
  - Single (1)
  - Twin (2)
  - Quad (4) / Quatro (pre multiprepínače)
  - Octo (8)
- počtu vstupov pre príjem z viacerých družíc:
  - Monoblock (1)
  - Monoblock twin (2)
- šumového čísla - čím nižšie, tým lepšie. Pre SD kvalitu je potrebné minimálne 0,3dB a pre HD sa odporúča 0,1dB.
- zosilnenia (zisku)
- polarizácie:
  - horizontálna
  - vertikálna
  - kruhová
- prenášaného pásma

Ďalšími menej dôležitými parametrami je spotreba prúdu, schopnosť spracovať obrazový formát MPEG2/MPEG4 prijímačom, alebo podpora DiSEqC ovládania.

Zdroj: [4]

[2] `http://www.satlex.net/sk/azel_calc.html`

[3] Vstupné frekvencie sa pohybujú v rozmedzí 10.7-11.7 a 11.7-12.75 GHz a sú prevádzané do 950-2150 MHz

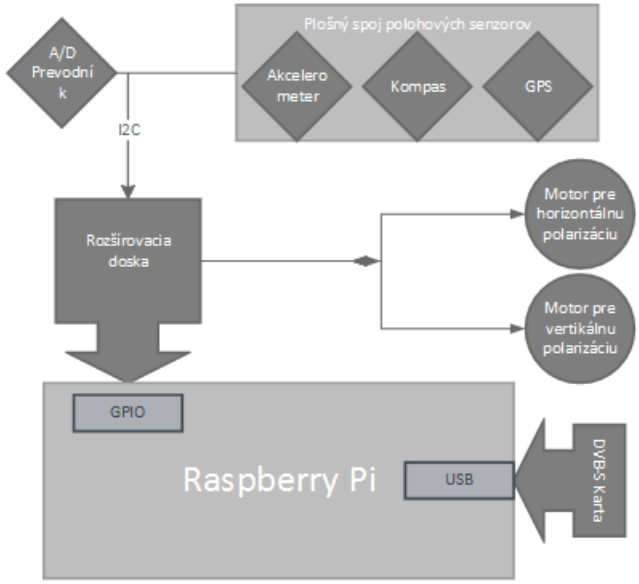
3 Konštrukcia mobilnej satelitnej antény

V tejto kapitole bude opísané, čo je potrebné na zostavenie mobilnej satelitnej antény. Jej cena sa pohybuje na trhu v rozmedzí 600 - 1000 $ (podľa hardvérového vybavenia). Najprv je potrebné zadefinovať, aké senzory budú nutné na orientáciu v priestore, alebo na výpočet elevácie a azimutu. Ďalej bude popísaná satelitná USB karta, prostredníctvom ktorej sa bude merať DVB-S signál.

Spracovanie dát zo senzorov a riadenie pohybových motorčekov bude mať na starosti ovládací kontrolér počítač Raspberry Pi. Je preto dôležité uviesť jeho architektúru s nutným programovým vybavením.

V závere kapitoly bude vysvetlená mechanická konštrukcia mobilnej satelitnej antény a ukázaná na obrázkoch praktická realizácia pohybu paraboly v plastovom kryte antény.

Celé hardvérové vybavenie spolu s anténou získam od vedúceho tejto práce, ktorý mi ho zapožičia komplet zapojené, ako ukazuje bloková schéma na obrázku 3.



Obr. 3: Bloková schéma zapojenia komponentov (zdroj vlastný)

3.1 Potrebné senzory

Spolu s rastúcim trendom internet veci sa senzory nájdu v takmer každom chytrom elektrickom zariadení. Pôvodne jednoduché zariadenia boli doplnené o „smart“ funkcie. Či už šlo o chladničky, pračky, alebo hrnčeky s teplotným čidlom.

Vznikli celé bezdrôtové senzorové siete (WSN), v ktorých sa senzory správajú autonómne. Ich vlastnosti môžu byť charakterizované ako:

- Samo-konfigurácia, samo-opravovanie, samo-optimalizácia a vlastná ochrana.
- Komunikácia na krátku vzdialenosť a multi-hop smerovanie.

- Husté nasadenie a kooperatívne úsilie uzlov.
- Často meniaca sa topológia v dôsledku zlyhania uzlov.
- Prísne obmedzenie v oblasti energetickej kapacity, výpočtového a vysielacieho výkonu.

K postaveniu mobilnej satelitnej antény však postačuje päť senzorov a to konkrétne kompas, akcelerometer, GPS senzor a dopĺňa ich A/D prevodník s DVB-S kartou. **Kompas** bude využitý pri nastavení azimutu a **akcelerometer** pri nastavení elevácie. **GPS senzor** bude potrebný na zistenie aktuálnej polohy pozemnej prijímacej antény, aby sa z nej dalo vypočítať smer natočenia k družici. Pri zmene azimutu je potrebné sledovať aktuálny uhol natočenia a podľa neho prispôsobovať ďalší smer točenia. To bude zachytávať **A/D prevodník** prostredníctvom odporového deliča. Podrobnejšie budú rozvedené v nasledovných podkapitolách.

3.2 Použité senzory

Budú použité modernejšie senzory oproti senzorom z mobilnej satelitnej antény, z ktorej konštrukcia vychádza. Od vedúceho tejto práce dostanem k dispozícií na implementáciu tieto senzory:

- **kompas** - HMC6352
- **akcelerometer** - BMA180
- **A/D prevodník** - MCP3221
- **GPS senzor** - LS23060
- **DVB-S karta** - TT-connect S-2400

3.2.1 Kompas

K nastaveniu azimutu je potrebné vedieť, kde je sever a jeho odklon od neho. Jedným z digitálnych kompasov, ktoré komunikujú prostredníctvom I2C zbernice, je **HMC6352** od firmy Honeywell. Používa sa v mnohých smartfónoch, v navigáciách a má teda široké zastúpenie v spotrebnej elektronike. Tento modul kombinuje dve osy magneto-odporových senzorov s podporou analógovo-digitálnych okruhov a výpočet smerovacieho algoritmu. Potrebuje k tomu 2,7 - 5,2V a pri 3V 1mA.

Čidlo disponuje mnohými funkciami na získanie, kalibráciu a výpočet smerovacích dát, a preto mu výrobca stanovil dva operačno-kontrolné módy:

1. **Prevádzkový mód** - definuje činnosť senzora.
2. **Výstupový mód** - definuje formát výstupných dát, ktoré môžu byť vo forme stupňov odchýlených od severného pólu, alebo vo forme „surových“ dát z magneto-odporových senzorov.

Pred akoukoľvek akciou nieje nevyhnutné senzoru poslať príkaz na konkretizáciu úlohy, ktorú budeme od neho požadovať. Správanie čidla je možné ovplyvniť zmenou režimu v akom pracuje. Senzor môže pracovať v troch prevádzkových režimoch:

1. **Pohotovostný mód** - štandardný mód, senzor čaká na príkazy a vykoná vždy iba jeden.
2. **Vyhľadávací mód** - procesor senzora očakáva príkaz na získanie dát, ktoré sa neustále načítavajú a po prijatí príkazu sa iba odovzdajú. Výhodou je menšia latencia a ihneď získané dáta.
3. **Neprerušovaný mód** - vykonáva kontinuálne meranie dát vo zvolenej frekvencií a posiela ich na výstup. Príkaz na získanie dát nieje potrebný. Je vhodný pre dátovo náročné aplikácie.

Budem využívať od výroby prednastavený pohotovostný mód a prepnutím do funkcie čítania dát cez tzv. „A command" s HEX hodnotou $0x41$ získam tak namerané **šestnásť bitové dáta.**

Pre nadobudnutie korektného azimutu je potrebné prispôsobiť vzdialenosť od severného pólu pre žiarič antény a samotného senzoru. Preto je nutné posunúť vrátenú hodnotu od senzora o $180°$ programovo, nakoľko senzor smeruje k severu z opačnej strany ako žiarič.

Uvedený senzor kompasu komunikuje prostredníctvom protokolu I2C na adrese *0x21* pripojenej na zbernici 1. Táto zbernica je však detailnejšie opísaná v podkapitole 4.1 na strane 23.

Zdroj: [6]

3.2.2 Akcelerometer

Na meranie elevácie je použité čidlo **BMA180** slúžiace na meranie stacionárneho zemského zrýchlenia, ale je vhodné aj pre meranie dynamických zrýchlení ako sú vibrácie a zrýchlenia pri pohybujúcich sa objektoch. Má v sebe integrované aj teplotné čidlo, takže vieme zistiť teplotu prostredia.

Vyznačuje sa veľmi nízkym šumom, spotrebou a širokým rozsahom merania $\pm 16g$. Napätie, ktoré potrebuje je 1,2 - 3,6V a $650\ \mu$A.

Výstupom digitálneho akcelerometra od firmy BOSH sú zrýchlenia troch osí v **štrnásť bitovom formáte**, ale pre výpočet náklonu stačia osi x a z, z ktorých je možné vypočítať požadovaný uhol α. Výsledný vzorec vyzerá nasledovne:

$$\alpha = \frac{(\arctan x / \arctan z) * 180}{\pi}$$

Vzhľadom na to, že čidlo bude umiestnené na plošnej doske, ktorá sa nachádza na zadnej vrchnej časti paraboly, bude ešte nutné upraviť prepočet, aby bol prispôsobený k žiariču a nie k polohe dosky. Z toho dôvodu vzniká takýto vzťah:

$$\beta = (\alpha * (-1)) + 40$$

Násobenie -1 je potrebné, pretože pohyb žiariča je inverzný voči pohybu dosky, tj. žiarič ide smerom hore, keď akcelerometer klesá. Konštanta -40 je odmeraná ako rozdiel uhla náklonu medzi žiaričom a plošnou doskou, kde budú umiestnené senzory.

Uvedený senzor akcelerometra komunikuje prostredníctvom protokolu I2C na adrese *0x40* pripojenej na zbernici 1. Táto zbernica je však detailnejšie opísaná v podkapitole 4.1 na strane 23.

Zdroj: [7]

3.2.3 A/D prevodník

Konštrukcia zapožičanej antény neumožňuje natáčať parabolu o $360°$ ktorýmkoľvek smerom. Preto som nútený sledovať v akej momentálnej relatívnej pozícií sa parabola nachádza. Táto informácia by obvykle pochádzala z krokového motorčeka, ktorý však nebudem mať k dispozícií. Táto vlastnosť bude nahradená kombináciou potenciometra a analógovo-digitálneho prevodníka **MCP3221** od firmy Microchip.

MCP3221 má jeden analógový vstup (Ain), na ktorom musí byť maximálne rovnaké napätie ako to, čo napája samotný prevodník. Na tento vstup sa privedie napätie U, ktoré A/D prevodník interpretuje **dvanásť bitovou hodnotou**. Veľkosť napätia U je priamo závislá od hodnoty odporu R [4], ktorá je menená potenciometrom (odporový delič) pri zmene polohy paraboly. Výška napätia U sa bude lineárne meniť v rozsahu 0 - 5V.

Výhodou A/D prevodníka je nízka spotreba prúdu. Pri napájacom napätí 2,7 - 5,5V je to maximálny prúd 250 μA. Ďalšou výhodou je rozlíšenie ôsmich zariadení pod jednou I2C adresou zbernice.

Pri dvanásť bitových hodnotách sa je možné dostať k maximálnemu číslu $2^{12} = 4096$, čo však vôbec nieje potrebné, pretože parabola sa pohybuje v kruhu, čo má maximálne $360°$. Z toho dôvodu si výstupnú hodnotu z A/D prevodníka upravím bitovým posunom o 3 miesta doprava na maximálne číslo $2^9 = 512$ a vydelím koeficientom 1,15, kde sa už dostanem na požadovanú maximálnu hodnotu $360°$. Vďaka tomu budem získavať relevantnejšie výsledky.

Uvedený senzor A/D prevodníka komunikuje prostredníctvom protokolu I2C na adrese *0x4d* pripojenej na zbernici 1. Táto zbernica je však detailnejšie opísaná v podkapitole 4.1 na strane 23.

Zdroj: [8]

3.2.4 GPS

Základom smerovania k družici Astra je zistenie polohy samotnej pozemnej antény, ktorá chce s ňou komunikovať a na základe nej natočiť parabolu správnym smerom. Na to poslúži GPS senzor **LS23060** od spoločnosti LOCOSYS.

Skladá sa z prijímacej antény, baterky a prijímacích okruhov GPS. Jeho výhodou je rýchle nájdenie navigačných satelitov, ktorých môže byt až 16 naraz a zmenu pozície vie identifikovať do sekundy pri nízkej spotrebe 3,3V a 41 μA. Poskytuje vynikajúcu citlivosť i v husto osídlených oblastiach, takže spĺňa požiadavky pre použitie v automobiloch.

[4] Ohmov zákon $U = I * R$

O korektnom získaní polohy z GPS satelitov vizuálne informuje blikanie LED diódy, čo po zapnutí napájania typicky trvá 42 sekúnd pri priamej viditeľnosti oblohy s presnosťou 2,5 metra.

GPS prijímač je poháňaný čipom ATMEL ANTARIS štvrtej generácie a jeho výstupom sú NMEA vety, ktoré sú podrobnejšie rozobraté v podkapitole 4.2. Pred štartom komunikácie s prijímačom je nutné nastaviť sériovú zbernicu nasledovnými hodnotami:

- 57 600 bps, štandardne 9 600 bps
- 8 dátových bitov
- žiadna parita
- 1 stop bit

Viac o senzore je možné sa dočítať v technickej dokumentácií dostupnej na adrese [5].

GPS senzor bude dostupný v systéme Linux cez sériové rozhranie na adrese `/dev/ttyAMA0`.

Zdroj: [9]

3.2.5 DVB-S karta

Pre presné naladenie kvalitného obrazu je potrebné poznať aj kvalitu prijímaného satelitného signálu. K jeho najlepšej hodnote sa bude nakoniec prispôsobovať smerovanie paraboly. Preto je zvolená DVB-S karta, ktorá si rozumie s Linuxom a z nej sa bude získavať sila signálu. Bude vybratý model **TT-connect S-2400** od výrobcu TechnoTrend.

Pripája sa prostredníctvom rozhrania USB 2.0 a potrebuje externé napájanie vo veľkosti 12V pri 1,7A, kedže takéto nedostane z USB portu.

Karta disponuje štandardnými funkciami potrebnými na zobrazovanie DVB-S obrazu na monitore počítača, ako napríklad MPEG2 kódovanie obrazu, EPG (elektronický programový sprievodca), alebo AC3 dekódovanie zvuku. Avšak najdôležitejšia je podpora otvoreného programového API, ktoré umožňuje vývojárom vytvoriť Linuxové ovládače. Po nahratí ovládačov *dvb-usb-tt-s2400-01.fw* do priečinka `/lib/firmware/` bude karta pripravená na použitie. Tieto ovládače sú dostupné online na adrese [6] a tiež sú zaradené do elektronickej prílohy tejto práce.

Karta umožňuje napájať LNB, ale to nieje potrebné, nakoľko LNB bude napájané zo set-top boxu, ktorý prijíma a dekóduje obraz.

DVB-S karta bude dostupná v systéme Linux cez súborové rozhranie na adrese `/dev/dvb/adapter0/frontend0`, aj keď sa pripája cez rozhranie USB.

Zdroj: [10]

[5] `http://www.nextboat.it/menu/it/accessori/GPS/locosys/LS2306x_datasheet_v1.0.pdf`

[6] https://github.com/OpenELEC/dvb-firmware/blob/master/firmware/dvb-usb-tt-s2400-01.fw

3.3 Ovládací kontrolér

Denne sa stretávame s výpočtovými zariadeniami, ktoré poskytujú širokú škálu funkcií na uľahčenie života, ale nie vždy je potrebná práve jednoduchosť. Pokiaľ by bola možnosť si vytvoriť taký nástroj, s ktorým by sa dalo kreatívne vytvárať a rozširovať jeho funkcionalitu, tak by bolo možné si ho prispôsobiť vlastným potrebám a presne na mieru. Aj za takýmto účelom bol stvorený mini počítač Raspberry PI (ďalej len RPi).

RPi môže poslúžiť pri ovládaní elektrických zariadení, ako zábavné mediálne centrum, alebo na bežné použitie pre kancelárske desktopové aplikácie. RPi myslelo aj na deti, ukazuje im príklad toho, že počítače nie sú len na hranie, kde sú pekné ikony na klikanie, ale že je ich možné využiť ako nástroj na tvorenie niečoho užitočného. Príkladom toho je jednoduchý programovací nástroj Scratch, ktorý má už predinštalovaný v operačnom systéme. Scratch je software s grafickým rozhraním, kde sa logika programu nastavuje presúvaním ikoniek na obrazovke.

RPi malo za cieľ dostať sa medzi širokú masu ľudí, ktorých práve bavia takéto stavebnice, kde je elektronika spojená s programovaním a za nízku cenu sa im to aj podarilo.

V nasledujúcich častiach je opísané hardvérové a softvérové vybavenie RPi, ktorý bude slúžiť ako ovládací kontrolér svojich periférií.

3.3.1 Hardvérové vybavenie

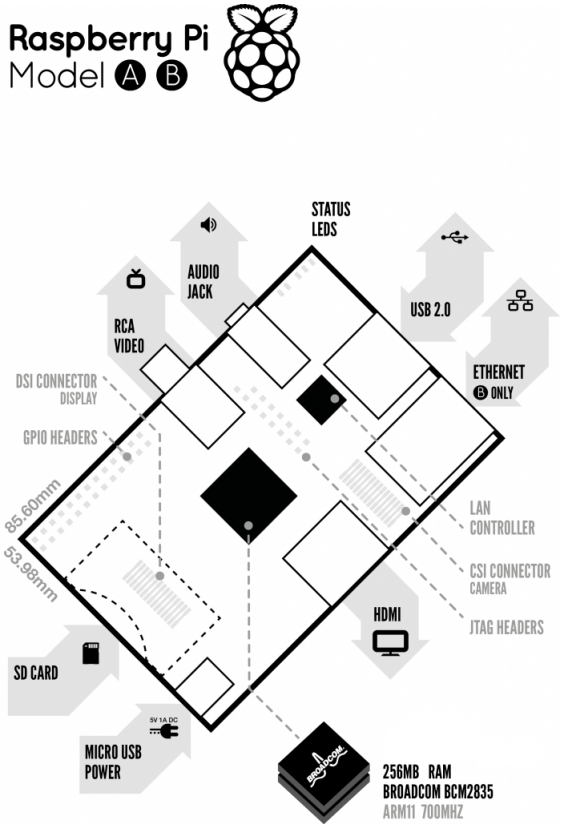
Do počítača o veľkosti kreditnej karty sa muselo vložiť také vybavenie, aby to zvládalo prehrávať video s vysokým rozlíšením, ale zároveň, aby bol energeticky úsporný.

Aritmetická logická jednotka, výpočtová jednotka a multimediálna jednotka sa zapúzdrili do jedného SoC riešenia. Tento jednotný komponent vykoná grafické výpočty s integrovanou **512 MB**[7] operačnou pamäťou a riadi komunikáciu s ďalším hardvérom. Používa inštrukčnú sadu známu ako ARM, konkrétne ARMv6[8] a jeho frekvencia je **700MHz**, s jedným jadrom.

Jednoduchší pohľad rozmiestnenia komponentov na plošnej doske RPi v.1 znázorňuje nasledovný obrázok 4.

[7] Verzia RPi 2 model B má 1 GB operačnej pamäte

[8] Verzia RPi 2 model B má inštrukčnú sadu ARMv7



Obr. 4: Diagram rozmiestnenia komponentov na RPi v.1 (zdroj elinux.org)

Ako je vidieť na vyššie uvedenom obrázku 4, tak doska počítača neobsahuje štandardný samec pre napájací konektor, ale napájanie je riešené prostredníctvom mikro USB konektora so stabilným napätím 5V a odporúčaným prúdom 1A. V prípade menšieho prúdu nemusia fungovať korektne USB periférie.

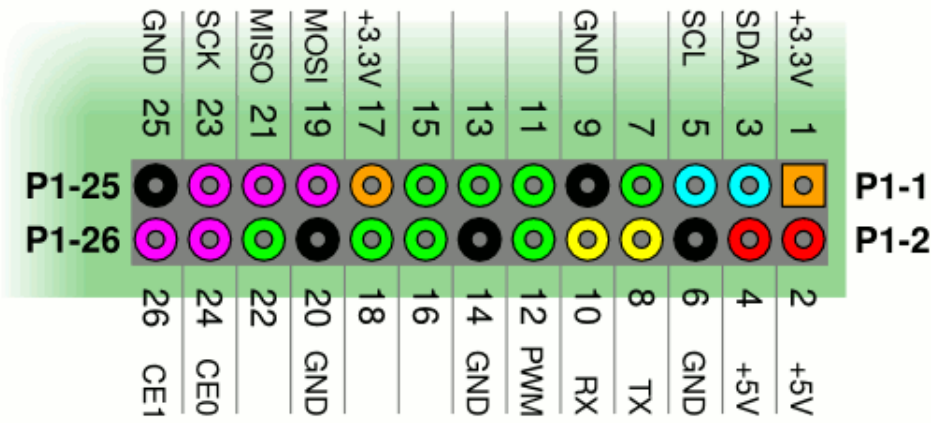
Zdroj: [11]

K hardvérovému vybaveniu neodmysliteľne patrí aj **všeobecné vstupno-výstupné rozhranie GPIO**, ktorého správanie je možné programovo ovládať pomocou softvéru. Prostredníctvom rozhrania GPIO sa k RPi pripájajú rôzne periférne zariadenia a rozširujúce dosky (napr. RPI Gertboard) a cez CPU sú ovládané.

Cez GPIO budem mať napojenú vlastnú rozširovaciu plošnú dosku na pripojenie konektorov externých senzorov a ďalších potrebných elektronických súčiastok, ako napríklad kondenzátory a stabilizátory napätia pre napájacie obvody elektrických motorčekov.[9] (viac v priloženej schéme). Aby sa obe dosky zmestili tesne k sebe a zaberali minimum miesta, tak **je nutné odstrániť S/PDIF a RCA video konektor** z dosky RPi, ktoré nie sú v použitej implementácií vôbec potrebné.

[9]Napájacie obvody pre elektrické motorčeky ide nahradiť plošnou doskou 4A Motor Shield `http://www.seeedstudio.com/depot/4A-Motor-Shield-p-1954.html`

GPIO konektor (angl. header) pre RPi v.1 poskytuje 26 pinov (2 rady po 13)[10]. Skutočné rozloženie GPIO pinov je možné vidieť na nasledovnom obrázku 5.



Obr. 5: Označenie pinov GPIO na RPi v.1 (zdroj advamation.com)

Ako je vidieť na obrázku 5, piny sú farebne rozlíšené a delia sa podľa účelu do niekoľkých skupín:

- **5V výstupné napätie** - červená farba.
- **3,3V výstupné napätie** - oranžová farba.
- **mínusové napätie** - čierna farba.
- **ovládanie vstupu a výstupu** - zelená farba.
- **rozhranie UART** - žltá farba.
- **rozhranie SPI** - fialová farba.
- **rozhranie I2C** - tyrkysová farba.

Ovládacie GPIO piny tolerujú maximálne vstupné napätie 3,3V, ktoré je aj na ich výstupe. Pri prepätí alebo pri statickom náboji hrozí trvalé poškodenie procesora, nakoľko tam nieje žiadna ochrana. Obvykle sa používajú tranzistory, zosilňovače alebo nejaký iný mechanizmus, ktorý vyrovnáva veľké prúdy. Softvérová ukážka použitia ovládacích GPIO pinov bude popísaná v podkapitole 6.1.

Rozhranie UART je určené na sériovú komunikáciu so sériovými zariadeniami.[11]

Sériovo periférne rozhranie (v skratke SPI) je určené na synchrónnu sériovú komunikáciu so SPI zariadeniami. Je to podobný koncept ako I2C, ale riadi sa inou normou.

[10] Plus verzie RPi majú 40 GPIO pinov

[11] Viac informácií o RPi sériovej komunikácií na web stránke `http://elinux.org/RPi_Serial_Connection`

Niektoré piny môžu používať pulzno-šírkovú moduláciu pre elektrické ovládanie zariadení, alebo pulzno-pozičnú moduláciu na ovládanie servo motorov.[12]

Rozhranie I2C umožňuje pripojiť hardvérové moduly iba dvoma kontrolnými vodičmi. Zbernica I2C bude podrobnejšie vysvetlená v podkapitole 4.1.

Zdroj: [12]

3.3.2 Softvérové vybavenie

Zadané požiadavky na operačný systém počítača RPi boli nízka cena, otvorenosť zdrojového kódu a podpora ARM architektúry. Toto jedine spĺňal operačný systém Linux a jeho distribúcie Debian, Fedora Remix a Arch Linux.

Inštalácia OS bude prebiehať podľa návodu uvedeného v knihe [11], kde preferujem konfiguráciu s vyhradenou väčšou pamäťou RAM pre procesor a to konkrétne 224MB. Grafický procesor zabral zvyšok operačnej pamäte. Konfigurácia sa upraví príkazom:

```
sudo cp /boot/arm224_start.elf /boot/start.elf
```

Bude uprednostnená rýchlosť počítača pred jeho grafickým zrýchlením z toho dôvodu, že s RPi sa normálne nebude pracovať cez monitor a video výstup, ale cez vzdialenú terminálovú konzolu (SSH) bez grafického prostredia, ktorej stačí minimum video pamäte a to len v prípade servisného zásahu.

Ďalší popis softvérového vybavenia (modulov) bude rozobratý v implementačnej časti tejto práce a to v kapitole 6 a v jej podkapitolách.

3.4 Mechanická konštrukcia

Konštrukcia zapožičanej mobilnej satelitnej antény vychádza z inšpirácie mobilnej satelitnej antény **Tailgater** od spoločnosti DISH. Akonáhle je nastavená, pripojí sa k DISH prijímaču, na správny satelit a môže sa začať používať. Jedine čo potrebuje je priama viditeľnosť na oblohu. Je odolná voči poveternostným vplyvom, odpudzuje dážď, vietor a UV žiarenie. Je dodávaná s držiakom, pomocou ktorého je možné anténu uzamknúť, aby nedošlo k jej odcudzeniu.

V prípade použitia satelitného servo motora by bolo natáčanie paraboly do požadovanej polohy riešené na základe riadiacich príkazov štandardu DiSEQc. S takýmto použitím satelitného motora by bolo dostupné široké spektrum satelitných pozícií, ktoré stačí raz nastaviť a neskôr sa o všetko postará prijímač v spolupráci s motorom na báze ovládacích príkazov DiSEQc. V tomto projekte si však vystačím s obyčajnými elektromotormi na 12V.

Zdroj: [13]

3.4.1 Praktická realizácia pohybu paraboly

Pevná anténa sa raz namontuje, nasmeruje, doladí, a potom s ňou už nieje potrebné manipulovať, až do jej opravy. Po nasmerovaní k družici a jemnom doladení sa zafixuje jej

[12] Viac informácií o SPI na web stránke `http://elinux.org/RPi_SPI`

držiak pomocou ktorého sa upevní k pevnému bodu, ako napríklad stožiar. Mobilná anténa sa musí vedieť natáčať na požadovaný smer v čase. Kedže konštrukcia paraboly nedovoľuje otáčanie okolo svojej osy viac ako o 360°, bude nutné sledovať, v akej relatívnej pozícií sa mechanizmus otáčania nachádza a k nemu prispôsobiť smer otáčania.

Zavediem preto dva pojmy pri nastavení azimutu:

- **Relatívny azimut** - tvorí uhol vodorovného otáčania, v ktorom sa nachádza parabola vzhľadom na konštrukciu antény. Na sledovanie tohto uhla bude použitý A/D prevodník opísaný v podkapitole 3.2.3.
- **Absolútny azimut** - tvorí skutočný uhol azimutu, ku ktorému sa parabola bude nastavovať. Na jeho sledovanie bude použitý kompas opísaný v podkapitole 3.2.1.

Preto uvediem štyri prípady relatívnych pozícií, do ktorých sa môže dostať anténa pri smerovaní na absolútny azimut.

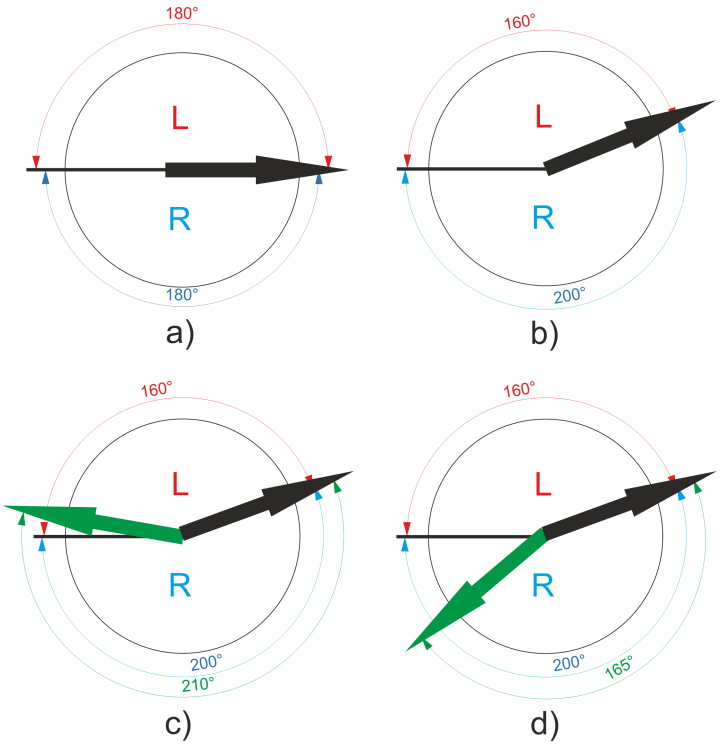
a) začiatočná poloha.

b) zmena relatívnej pozície zo začiatočnej polohy smerom doľava, alebo doprava.

c) zmena relatívnej pozície k absolútnej mimo vyhradeného maximálneho relatívneho posunu.

d) zmena relatívnej pozície k absolútnej v rámci vyhradeného maximálneho relatívneho posunu.

Pre lepšiu predstavivosť spôsobu smerovania antény uvádzam príklady z každej pozície, na konkrétnych príkladoch zobrazených na obrázku 6.



Obr. 6: Natáčanie paraboly (zdroj vlastný)

Na vyššie uvedenom obrázku 6 je možné vidieť štyri prípady relatívnych pozícií, ktoré už boli spomenuté. Vysvetlivky k použitým značkám sú v nasledovnej tabuľke:

Značka	**Účel**
L	Left (v prekl. ľavá) strana paraboly
R	Right (v prekl. pravá) strana paraboly
Stupne červenou farbou	maximálna možná zmena azimutu smerom doľava o dané stupne
Stupne modrou farbou	maximálna možná zmena azimutu smerom doprava o dané stupne
Stupne zelenou farbou	požadovaný pohyb smerom doprava o dané stupne
Čierny ukazovateľ	aktuálny relatívny smer azimutu
Zelený ukazovateľ	požadovaný absolútny smer azimutu

Tabuľka 1: Vysvetlivky k obrázku natáčania paraboly

Teraz bude podrobnejšie rozobratý každý jeden príklad z obrázku 6

Príklad 3.1
Pozícia a) znázorňuje začiatočnú absolútnu polohu, keď je anténa v strednej pozícii a je ju možné posunúť jedným alebo druhým smerom maximálnej o $180°$. ■

Príklad 3.2
Pozícia b) znázorňuje relatívnu pozíciu antény, keď sa nachádza na $160°$. V tejto polohe sa parabola môže otočiť smerom doľava o maximálne $160°$ alebo smerom doprava o $200°$. Treba na to dbať pri smerovaní paraboly za skutočným absolútnym azimutom. ■

Príklad 3.3
Pozícia c) znázorňuje relatívnu pozíciu antény, keď sa nachádza na $160°$, ako v predchádzajúcom príklade (tento uhol si označím ako α). Ďalej je vidieť, že potrebný skutočný azimut sa nachádza o $210°$ vpravo od relatívnej pozície (tento uhol si označím ako β).

Uhol β je vypočítaný ako rozdiel medzi aktuálnym a potrebným skutočným azimutom.

Ako je možné spozorovať $\beta > \alpha$, a to nedovoľuje sa posunúť vpravo o $210°$, keďže konštrukcia dovolí posun len o $200°$. Z toho dôvodu som nútený sa posúvať opačným smerom a nastaviť sa na relatívnu pozíciu z druhej strany pomocou vzťahu $360° - \beta = 360 - 210 = 150$. Z toho vyplýva, že sa mám posunúť doľava o $150°$ a dostanem sa na požadovaný absolútny azimut. ■

Príklad 3.4
Pozícia d) znázorňuje relatívnu pozíciu antény, keď sa nachádza na $160°$, ako v predchádzajúcom príklade (tento uhol si označím ako α). Ďalej je vidieť, že potrebný skutočný azimut sa nachádza o $165°$ vpravo od relatívnej pozície (tento uhol si označím ako β).

Uhol β je vypočítaný ako rozdiel medzi aktuálnym a potrebným skutočným azimutom.

Ako je možné spozorovať $\beta < \alpha$, a preto je dovolené sa posunúť vpravo o $165°$, keďže konštrukcia dovoľuje posun do $200°$. ■

Vyššie uvedené praktické príklady poukazujú na správanie antény, keď by bola pootočená viac v ľavej časti. V prípade, že by sa nachádzala viac v pravej časti (tj. relatívny azimut α by bol väčší ako $180°$), tak by bolo správanie úplne rovnaké, len by boli opačné smery otáčania a ľavá strana sa zamení za pravú.

4 Komunikácia periférii

Po tom, ako bolo fyzicky navrhnuté elektrické zapojenie senzorov, ovládací kontrolér a mechanická konštrukcia, je možné definovať ich spôsob komunikácie, ako budú medzi sebou jednotlivé senzory komunikovať.

4.1 Zbernica I2C

I2C bus je skratka, ktorá vznikla z IIC bus, teda Internal-Integrated-Circuit Bus. Ako už názov napovedá, jedná sa o internú dátovú zbernicu slúžiacu pre komunikáciu a prenos dát medzi jednotlivými integrovanými obvodmi, väčšinou v rámci jedného zariadenia. Vyvinula ju firma Philips približne pred 20 rokmi a odvtedy prešla niekoľkými vylepšeniami.

V dnešnej dobe tento spôsob „komunikácie“ podporuje rad integrovaných obvodov nielen od firmy Philips. Jedná sa predovšetkým o mikrokontroléry, sériové pamäte, inteligentné LCD, audio a video obvody, A/D a D/A prevodníky a niektoré ďalšie digitálne riadené obvody. Hlavnou výhodou je, že obojsmerný prenos prebieha len po dvoch vodičoch:

- **dáta SDA (serial data)** - slúži pre prenos sériových dát.
- **hodiny SCL (serial clock)** - slúži pre prenos hodinového signálu.

Predovšetkým u mikrokontrolérov to výrazne optimalizuje nároky na počet vstupno-výstupných pinov a celkovo zjednodušuje výsledné zapojenie. Na jednu zbernicu môže byť pripojených viac integrovaných obvodov. V základnej verzii sú obvody adresované siedmimi bitmi a v rozšírenej verzii desiatimi. To umožňuje pripojenie **128 respektíve 1024 čipov** s rôznou adresou na jednu spoločnú zbernicu. V praxi sú však tieto čísla podstatne nižšie, pretože adresa čipu sa väčšinou nedá určiť plnými siedmimi, alebo desiatimi bitmi, ale napríklad len tromi. Niekedy nie je možné určiť adresu vôbec a je daná na pevno pre daný typ čipu. Takých čipov teda na jednej zbernici nemôže byť viac ako jeden.

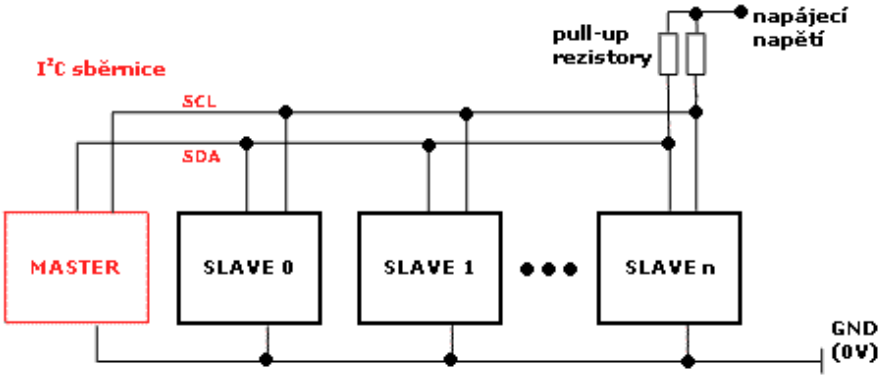
Prenosová rýchlosť zbernice je pre väčšinu aplikácií dostatočná aj v základnej verzií, kde je **frekvencia hodín 100 kHz.** Vo vylepšených verziách to môže byť **400 kHz alebo 1 MHz**, ale nie všetky integrované obvody túto verziu podporujú. Rýchlosť prenosu potom musí byť prispôsobená pochopiteľne „najpomalšiemu“ čipu na zbernici.

Obidva vodiče musia byť implicitne v logickej jednotke a to je zaistené tzv. pull-up rezistormi, ktoré sú pripojené medzi vodičom a napájacím napätím. Ich odpory majú hodnotu v rádoch jednotiek kilo-ohmov. Čím je vyššia komunikačná frekvencia, tým musia byť nižšie hodnoty týchto odporov. [13]

Týmto spôsobom je zabezpečená práca liniek SDA a SCL v obidvoch smeroch. Pokiaľ by došlo ku kolízii (chceli by vysielať viaceré obvody naraz), poškodili by sa iba úrovne signálu a nie vysielacie obvody. Spätnou väzbou je zaistené, že obvod môže pracovať aj ako vysielač, aj ako prijímač .

[13] Pre 100kHz postačuje 4K7

Možné prepojenie jednotlivých integrovaných obvodov ukazuje nasledujúci obrázok 7.



Obr. 7: Možné blokové zapojenia I2C čipov (zdroj fl.hw.cz)

4.1.1 Komunikačné režimy

Integrované obvody môžu spolu komunikovať v dvoch režimoch:

1. **master - slave** - väčšinou mikrokontrolér je nastavený ako master a všetky ostatné obvody sú ako slave.
2. **master - master** - tzv. multi-master, kedy je čipov na zbernici v režime master niekoľko.

Master pri akomkoľvek prenose generuje hodinový signál na vodiči SCL. Keď začne vysielať, tak všetky ostatné čipy prijímajú a len podľa adresy je určené, ktoré dáta majú prijať. Čip, ktorý chce vysielať, alebo prijímať dáta, musí najprv definovať adresu čipu, s ktorým chce komunikovať a či pôjde o čítanie alebo zápis. To určuje **najnižší read / write bit**, ktorý je súčasťou adresy.

- **R/W = 1** - čip sa správa ako vysielač (čítame z neho dáta).
- **R/W = 0** - čip sa správa ako prijímač (zapisujeme do neho dáta).

Obvody, ktoré pracujú v oboch smeroch rozlišujú dve adresy a adresa pre vysielanie je o jednotku vyššia než pre príjem dát. V kapitole 3.2 na strane 12 bol opísaný každý jeden použitý senzor s pridelenou I2C adresu. Pre pripomenutie sú zhrnuté v nasledovnej tabuľke 2.

Účel	Model	I2C adresa [HEX]
Kompas	HMC6352	0x21
Akcelerometer	BMA180	0x40
A/D prevodník	MCP3221	0x4d

Tabuľka 2: Mapovanie I2C adries

Implementácia obsluhy integrovaných I2C obvodov je riešená v systéme Linux súborovým rozhraním `/dev/i2c-x`, kde `x` predstavuje adresu zbernice. Tieto obvody sú ovládané štandardnými I2C nástrojmi, ktoré budú rozpísané v kapitole 6.1 na strane 34.

4.1.2 Princíp prenosu

Princíp prenosu je zhrnutý do nasledovných faktov.

4.1.2.1 Pokojový stav Je zaistený logickými jednotkami na oboch vodičoch. V tom prípade master negeneruje hodinový signál a neprebieha žiadny prenos. Logické jednotky sú na oboch vodičoch zaistené pull-up rezistormi, takže pokojový stav nastane aj pokiaľ sú výstupy obvodu master v stave vysokej impedancie (odpojené).

4.1.2.2 Štart bit Iniciuje prenos alebo jeho ďalšiu časť. Je vygenerovaný tak, že sa zmení úroveň SDA z logickej 1 na 0 zatiaľ čo je SCL v logickej 1.

4.1.2.3 Prenos dát Dáta sú prenášané po jednom byte, čiže 8 po sebe idúcich bitov, od najvyššieho po najnižší. Pri prenose dát sa môže logická úroveň na SDA meniť iba ak je SCL v logickej 0. Pri každom pulze na SCL je prenesený jeden bit.

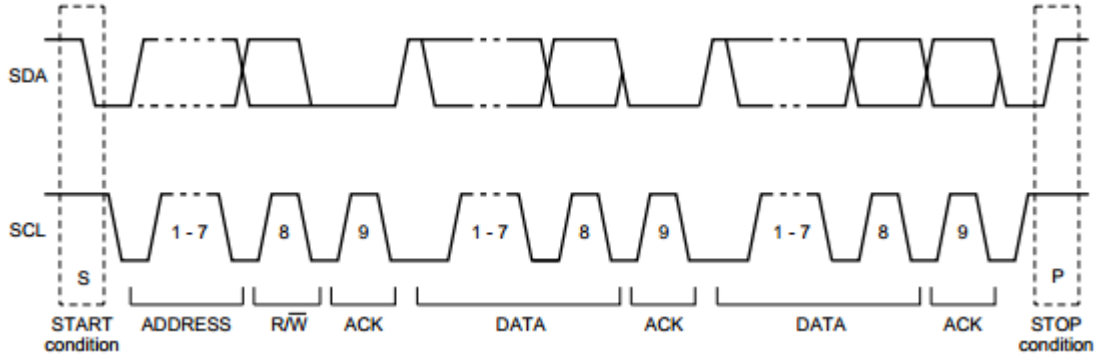
4.1.2.4 Potvrdzujúci bit acknowledge (v skratke Ack) Tento bit slúži na potvrdenie správneho prijatia dát. Ack bit sa odosiela rovnakým spôsobom, ako by sa odosielal deviaty bit dát, ale s tým rozdielom, že ho generuje čip, ktorý prijímal (prijímač) a nie ten, ktorý dáta odosielal.

Ak prenos prebehol v poriadku tak odošle logickú 0. Logická 0 potvrdzujúceho bitu znamená tiež to, že je prijímač pripravený na príjem ďalšieho bytu, ktorý nasleduje okamžite po ňom, pri ďalšom pulze na SCL.

Ak prenos zlyhal odošle logickú 1. Alebo ak má dôjsť k ukončeniu prenosu, tak neodošle nič. Pull-up rezistor potom zabezpečí, že bude na SDA logická 1 a Ack bit (v logickej 0).

4.1.2.5 Stop bit Ukončuje prenos. Je vygenerovaný podobne ako štart bit. Logická úroveň SDA sa zmení z 0 na 1, kým je SCL v logickej 1. Stop bit môže byť generovaný len po „nepotvrdenom prenose“, teda len po prijatí Ack v logickej 1.

Celý prenos je vyobrazený na nasledovnom obrázku 8.



Obr. 8: Kompletný I2C dátový prenos (zdroj nxp.com)

Predovšetkým pri vyšších prenosových rýchlostiach, respektíve pri hodinových frekvenciách 400 kHz a 1 MHz a dlhších vodičoch SCL a SDA by mohlo pri nesprávnom návrhu plošného spoja alebo celej konštrukcie dochádzať k rušeniu a ku chybám v prenose.

Okrem potvrdzujúceho bitu Ack nie je prenos nijako kontrolovaný. Bit Ack potvrdzuje iba prenesenie každého jedného bytu dát, ale nie to, či boli dáta prenesené správne. Preto je vhodné, aby boli vodiče SDA a SCL čo najkratšie a aby sa v ich blízkosti nevyskytovali výkonne alebo vysokofrekvenčné obvody.

Zdroj:[14, 15]

4.2 GPS štandard NMEA 0183

Americká spoločnosť National Marine Electronic Association (v skratke NMEA) špecifikovala štandard na prenášaný elektrický signál medzi rozhraniami námorných elektrických zariadení a počítačmi.

Štandard NMEA 0183 tvorí špecifický formát viet, ktoré očakávajú GPS prijímače a ich programy na určovanie polohy v reálnom čase. Tieto dáta zahŕňajú kompletné informácie o polohe, rýchlosti, čase a sú spracovávané GPS prijímačmi. Hlavným účelom NMEA bolo kategorizovať tieto dáta do viet, ktoré sú navzájom sebestačné, nezávislé a používané inou kategóriou zariadení.

Všetky štandardné vety začínajú prefixom, ktorý definuje typ zariadenia, ktorý používa tento typ vety. Pre všeobecné GPS prijímače je **prefix GP**, za ktorým nasleduje sekvencia troch písmen, ktoré definujú obsah viet. Okrem spoločnosti NMEA vyrábajú navigačné zariadenia aj iné firmy ako napríklad Garmin, ktoré dopĺňajú štandardné NMEA vety o svoje proprietárne, kde doplňujú vlastné dodatočné dáta. V tom prípade sa NMEA vety **začínajú písmenom P** a ďalšie 3 znaky tvoria identifikátor firmy.

Charakteristiku NMEA viet je možné zhrnúť do niekoľkých bodov:

- začínajú znakom „ $ “.
- maximálna dĺžka vety je 80 znakov.

- vetu tvoria ASCII znaky umiestnené na jednom riadku.
- hodnoty vo vete sú oddelené čiarkou.
- koniec tvorí kontrolný súčet v hexadecimálnom tvare, ktorý sa identifikuje znakom „ * “. Reprezentuje operáciu XOR celej vety až po jej začiatok, bez znaku „ $ “.

4.2.1 Komunikačné rozhranie

Komunikačné rozhranie GPS zariadení je navrhnuté tak, aby spĺňalo požiadavky NMEA protokolu. Väčšinou sú kompatibilné s počítačovými sériovými portami. Štandard NMEA po mnohých rokoch prešiel niekoľkými verziami [14], kde spočiatku komunikácia prebiehala prostredníctvom protokolu RS 232 a v roku 1992 prešla na RS422.

Pre bežné využitie NMEA štandardu v GPS prijímačoch sú potrebné iba dva vodiče:

- data out - údaje z GPS
- ground - zem

Tretí vodič (data in) by bol potrebný iba vtedy, keby bolo potrebné odosielať GPS prijímaču nejaké dáta, napríklad v podobe „waypoints“, alebo DGPS dáta.

Rýchlosť rozhrania pre NMEA je štandardne 4800 bit/s, s veľkosťou prenášaného slova 8 bitov, bez nastavenej parity a s nastavením jedného stop bitu. Pri tejto rýchlosti je možné prenášať 480 znakov za sekundu, z čoho vyplýva, že môžme preniesť 6 rôznych viet za sekundu. Preto je štandard NMEA navrhnutý pre neustálu komunikáciu na pozadí a softvérom sa rozhodne, ktoré dáta sa použijú.

Niektoré modely Garmin a ešte iné značky, umožňovali nastaviť rýchlosť na 9600 bit/s, alebo dokonca vyššiu, ale to sa odporúča iba v prípade, že 4800 bit/s funguje v poriadku, lebo to zvyšuje citlivosť programu.

4.2.2 Ukážka NMEA vety

Medzi najvýznamnejšie NMEA vety patrí GGA, ktorá poskytuje aktuálny fixačný status, súradnice lokácie a stav satelitných dát.

Rozbor GPGGA vety je ukázaná na nasledovnom príklade:
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

- **GPGGA** – je identifikátorom vety.
- **123519** – reprezentuje koordinovaný svetový čas (UTC), kedy bola odoslaná veta vo formáte hhmmss (hodiny minúty sekundy).
- **4807.038** - udáva zemepisnú šírku (angl. latitude) vo formáte ddmm.sss (stupne minúty sekundy). V tomto prípade $48°7'$ a $38''$.
- **N** - vyjadruje severnú časť pologule.

[14] aktuálna verzia NMEA 0183 štandardu je 4.1

- **01131.000** - udáva zemepisnú dĺžku (angl. longitude) vo formáte dddmm.sss (stupne minúty sekundy). V tomto prípade $11°31'$ a $000''$.
- **E** - vyjadruje východnú časť pologule.
- **1** - reprezentuje kvalitu GPS signálu a v tomto prípade je to GPS signál fixný (bez diferenčných korekcií).
- **08** - je počet viditeľných družíc.
- **0.9** - definuje horizontálne riadenie pozície.
- **545.4** - nadmorská výška.
- **M** - použitá jednotka nadmorskej výšky.
- **46.9** - výška geoidu nad elipsoidom WGS84.
- **M** - použitá jednotka výšky geoidu.
- _ - (prázdne pole), čas v sekundách od poslednej aktualizácie DGPS.
- _ - (prázdne pole), identifikátor stanice DGPS.
- ***47** - kontrolný súčet.

Toto je jediná veta, ktorá ukazuje nadmorskú výšku.
Zdroj: [16]

5 Návrh softvérového modelu

Po zdokumentovaní teórie geostacionárnych satelitov bolo možné navrhnúť konštrukciu mobilnej satelitnej antény s potrebnými senzormi na vnímanie fyzického prostredia. Tiež bolo vysvetlené, ako tieto periférie spolu komunikujú a ako sú ovládané mikro počítačom RPi, ktorý však potrebuje mať v sebe „programového ducha“. Program, ktorého cieľom bude natáčanie za družicou a bude to celé riadiť, bude navrhnutý v nasledujúcich podkapitolách.

Pri návrhu softvéru je dôležité zhrnúť požiadavky na projekt z používateľského a funkcionálneho hľadiska, a preto sa v nasledujúcej kapitole, ako prvé rozoberie špecifikácia požiadaviek na aplikáciu.

5.1 Špecifikácia požiadaviek

Účelom vyvíjaného programu bude získavať dáta z pripojených senzorov, ktoré bude následne spracovávať a vypočítavať z nich potrebný sklon a smer paraboly. Následne sa dostane do požadovanej pozície prostredníctvom pohybových funkcií a jemne doladí k najlepšiemu príjmu DVB-S satelitnému signálu.

Mechanizmus paraboly bude smerovaný k družici Astra 23,5 východne a bude sa polohovať podľa potrebnej:

- elevácie
- azimutu
- GPS polohy
- sily DVB-S signálu

Okrem potrebnej elevácie a azimutu sa bude načítavať aj ich aktuálna hodnota a po zmene polohy sa skontroluje, či sa parabola nenachádza v požadovanej polohe.

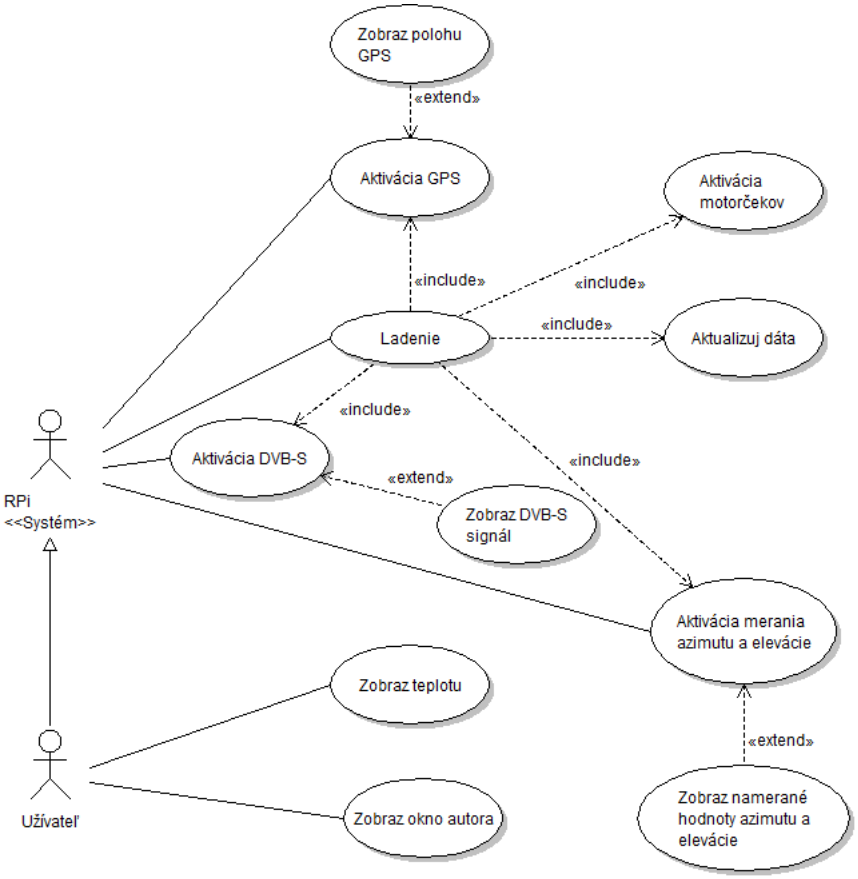
Po dosiahnutí požadovanej polohy sa aplikácia prepne na **jemný režim ladenia**, pri ktorom mení polohu v menšom rozsahu a pri každej zmene polohy si skontroluje hodnotu DVB-S signálu. Pri najväčšej sile signálu si zapamätá polohu, ku ktorej sa vráti po pár pokusoch.

Za účelom diagnostiky a kontroly smerovacích algoritmov sa budú vyššie spomenuté ukazovatele zobrazovať na monitorovacej konzole a prípadné chyby zaznamenávať na disk, nakoľko nieje stály obrazový výstup počítačového kontroléru antény a budú tak dostupné aj po vypnutí napájania antény.

5.2 Prípady použitia aplikácie

Na obrázku 9 uvádzam vymodelovaný diagram Use Case, ktorý zjednotí pohľad na funkcionalitu programu. Je to istý druh grafického zobrazenia funkčných požiadaviek.

V diagrame sa nachádzajú dve role, ktoré budú spúšťať jednotlivé prípady užitia. Prvoradý je samotný systém RPi, čiže automatická programová logika, ktorá spúšťa jednotlivé akcie a nieje nutný manuálny zásah užívateľa. Druhá rola je užívateľ, ktorý má možnosti ako RPi systém a k tomu si na vyžiadanie môže zobraziť teplotu okolia, alebo zobraziť informačné okno autora programu.



Obr. 9: Use Case diagram vyvíjaného programu (zdroj vlastný)

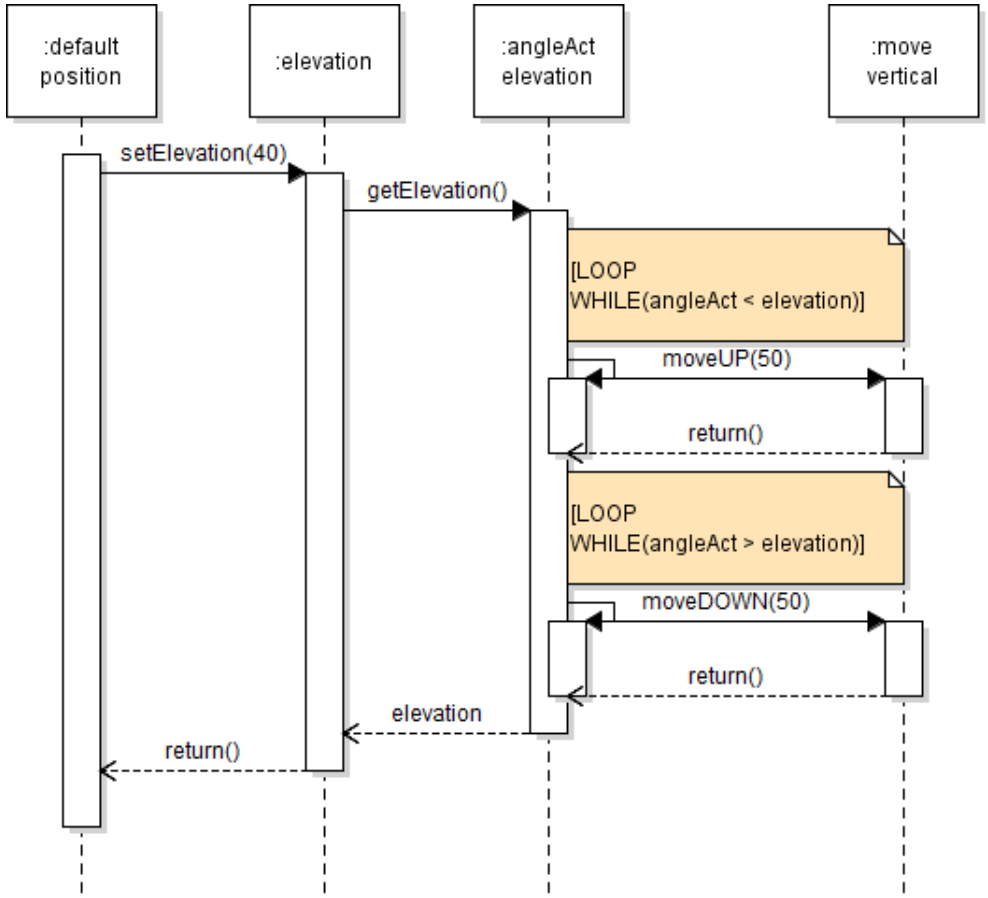
Najdôležitejší prípad použitia je „Ladenie“, ktorý v sebe zahŕňa ďalšie povinné prípady použitia (napríklad „Aktivácia GPS“), pretože funkciu ladenia nemôžme bez nich vykonať. Táto povinná väzba je označovaná pomocou šípkou include.

Ďalej si je možné všimnúť šípku extends, ktorou sa tiež označuje pridruženie prípadu použitia k inému prípadu použitia, ale táto väzba nieje povinná, nakoľko k jej volaniu nemusí dôjsť, ako napríklad „zobrazenie polohy GPS“.

5.3 Sekvenčný diagram východzej pozície

Ďalší z UML nástrojov sú sekvenčné diagramy, ktoré boli zvolené, aby sa presnejšie chápali požiadavky na dynamické správanie programov. Nasledovné dva obrázky predsta-

vujú správanie funkcie `setDefaultPos()`, ktorá nastaví polohu antény do východzej strednej pozície.

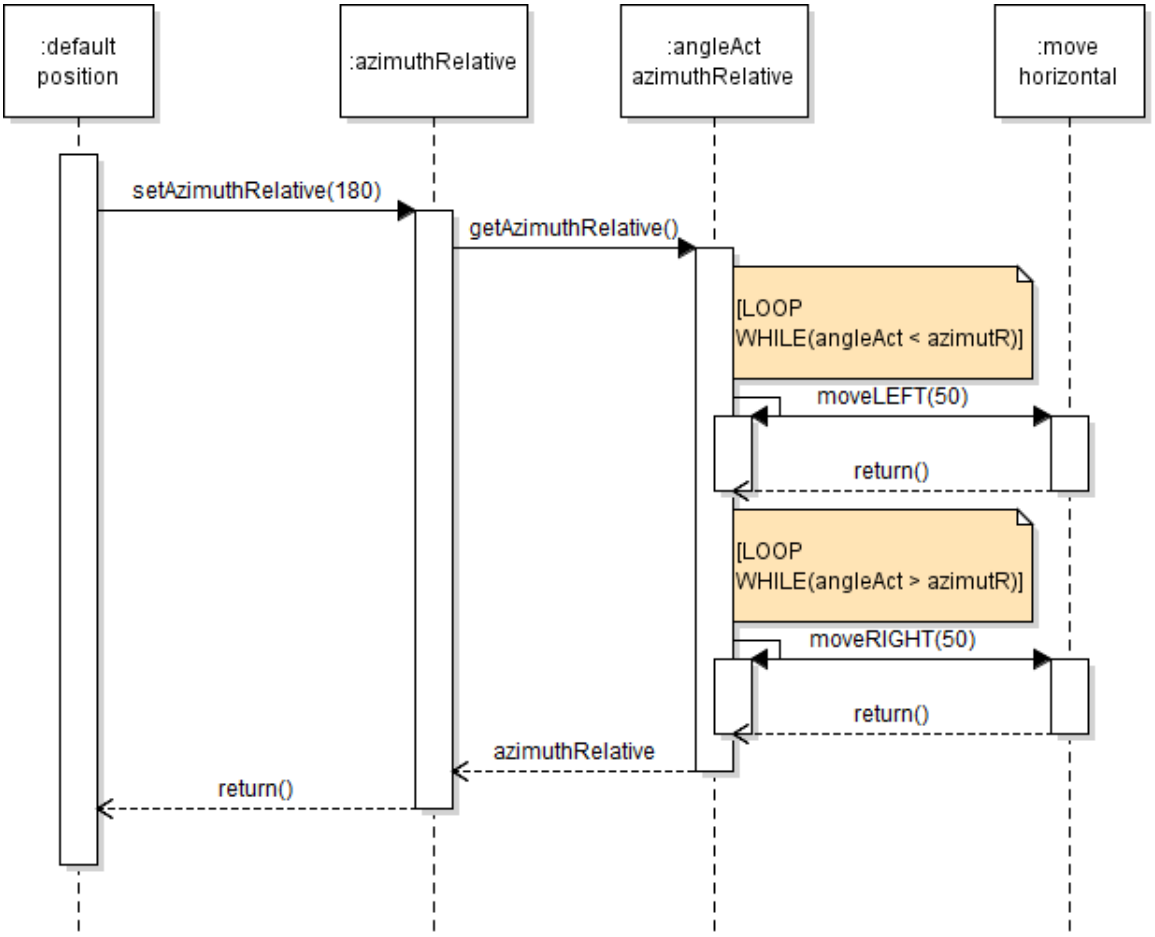


Obr. 10: Sekvenčný diagram nastavenia elevácie (zdroj vlastný)

Funkcia `setDefaultPos()` bude volať dve funkcie, jednu na nastavenie vertikálnej a druhú na horizontálnej polohy:

1. `setElevation(40)`
2. `setAzimuthRelative(180)`

Z dôvodu väčšej veľkosti obrázka bol diagram rozdelený na dva menšie, ktoré by boli inak pod sebou v jednom obrázku. Obrázok 10 predstavuje prvú volanú funkciu a obrázok 11 predstavuje druhú volanú funkciu.



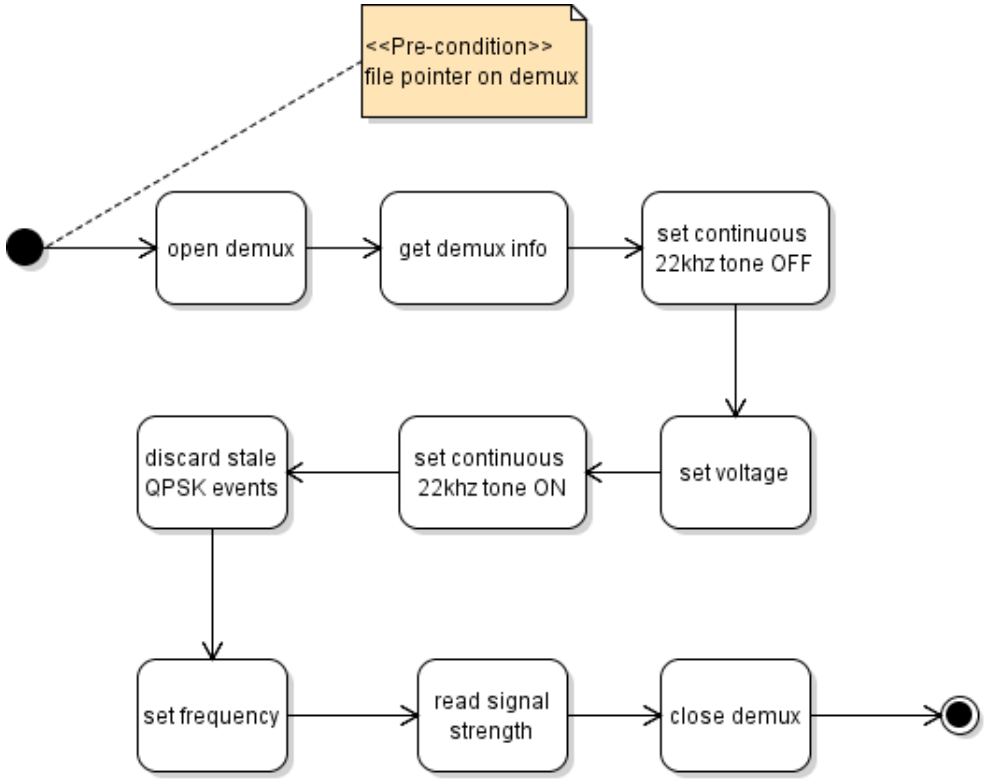
Obr. 11: Sekvenčný diagram nastavenia relatívneho azimutu (zdroj vlastný)

Diagramy modelujú GET a SET operácie nad elevaciou a relatívnym azimutom. V princípe ide o jednoduchú logiku, keď za každým pohybom antény, v dĺže trvania t (v našom prípade $t = 50ms$), sa skontroluje aktuálna hodnota premennej, ktorú je potrebné dosiahnuť. **Počiatočnú pozíciu tvorí** $elevacia = 40°$ a $azimutR = 180°$.

Rovnaké logické správanie, ako pri nastavení relatívneho azimutu, by mala mať funkcia `setAzimuthCompass()`, ktorá svoj uhol prispôsobuje aktuálnej hodnote kompasu.

5.4 Diagram aktivít pri použití DVB-S

Nasledovný obrázok 12 ilustruje aktivity, ktoré sú nevyhnutné pri čítaní dát z DVB-S karty. Podobne ako sekvenčný diagram, vyjadruje dynamický náhľad na softvér, v ktorom sú jednotlivé udalosti na seba závislé v čase a nemôže sa jedna aktivita vykonať skôr ako predchádzajúca.



Obr. 12: Aktivity diagram použitia DVB-S karty (zdroj vlastný)

Vyššie uvedený aktivity diagram hovorí, aké všetky aktivity je potrebné vykonať, aby som dostal DVB-S signál. Tieto operácie som si rozdelil do troch častí (funkcií), pretože nieje potrebné neustále zatvárať a otvárať demux zariadenie, keď čítam dáta v cykle.

Aktivity od *open demux* až po *set frequency* presuniem do funkcie `initializeDVBS()`, ktorú zavolám iba raz pri spustení. Následne môžem načítavať dáta až do ukončenia programu a uzatvorenia demux zariadenia. Dáta sú na požiadanie dostupné a je to tak rýchlejšie.

6 Implementácia

Táto kapitola je zameraná na technické detaily pri implementácií ukážkového satelitného programu, ktorý som si nazval **SatBerry**. Názov je odvodený od slova Sat, ako satelit a Berry, ako časť z názvu počítača Raspberry.

Popíšem konfiguračný postup operačného systému ovládacieho kontroléra a jeho modulov. Vysvetlím a ukážem najdôležitejšie funkcie a prácu s knižnicami. Následne budem rozoberať použité implementačné nástroje, ako programovací jazyk a vývojové prostredie a tiež niečo poviem o kríženej kompilácií.

Na záver kapitoly začnem ladiť signál s mojím vytvoreným programom, merať signál pomocou aplikácií tretích strán a testovať funkcionalitu jednotlivých modulov.

6.1 Konfigurácia Raspbian OS

Prvé kroky smerovali k inštalácií I2C nástrojov, ktoré sú k dispozícií z repozitárov zadaním príkazu: `apt-get install i2c-tools`

Po inštalácií bolo umožnené použitie rôznych nástrojov vzťahujúcich sa k protokolu I2C, ktoré zahŕňajú:

- **i2cdetect** - nástroj na prehľadávanie i2c zbernice.
- **i2cdump** - nástroj na vytiahnutie všetkých adries dostupných na danej zbernici.
- **i2cget** - nástroj na čítanie dát priamo z adresy zariadenia na určenej zbernice.
- **i2cset** - nástroj na zápis dát priamo do adresy zariadenia na určenej zbernici.

Zdroj:[17]

Ďalej som sa venoval I2C modulom a ich integráciou v systéme. Na samom začiatku je nutné povoliť moduly. Je potrebné postupovať podľa týchto krokov:

1. **aktivovať moduly** - príkazmi `modprobe i2c-dev` a `modprobe i2c-bcm2708`
2. **skontrolovať aktiváciu** - príkazom `lsmod`
3. **zaistiť ich aktiváciu po reštarte OS** - dopísaním ich do súboru */etc/modules*

Pre testovanie funkčnosti GPIO rozhrania bola použitá **utilita gpio**, dostupná z balička *wiringPi*. Postup, akým je možné nainštalovať tento balík je zverejnený na internetovej adrese [18]. Okrem testovacieho programu *gpio* balíček obsahuje aj hlavičkové súbory, ktoré je možné využiť pri vývoji programov v jazyku C pracujúcich s GPIO rozhraním. Avšak túto knižnicu som nepotreboval, lebo som pracoval s knižnicou BCM 2835, ktorú popíšem v podkapitole 6.3.5 na strane 41.

Na test GPIO rozhrania bola použitá nasledovná postupnosť príkazov:

1. **gpio** readall - jeho výstupom je tabuľka so všetkými sprístupnenými pinmi a ich aktuálnymi napäťovými hodnotami a režimom. Obsahuje vlastné označenie pinov a k nej prislúchajúce pôvodné označenie knižnicou a výrobcom BCM.

2. **gpio** mode 0 out - nastavený pin 0 do výstupného režimu.

3. **gpio** write 0 1 - poslané napätie pin 0.

K rovnakému výsledku sa dá dopracovať, keď sa bude komunikovať cez súborové rozhranie pinov `/sys/class/gpio/` zadaním nasledovných príkazov:

```
echo "0" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio7/direction
echo "0" > /sys/class/gpio/unexport
```

Automatickú aktualizáciu stavov všetkých pinov pri ich zobrazení bolo možné dosiahnuť zadaním príkazu `watch -n 0.5 "gpio readall"`.

Treba však upozorniť, že aplikácia *gpio* používa iné číslovanie pinov GPIO, ako samotné RPi a knižnica BCM a nie všetky piny sú dostupné prostredníctvom tejto knižnice. V nasledujúcej tabuľke 3 je ukázaný názorný rozdiel práve na využitých pinoch pre aktiváciu elekromotorčeka želaným smerom.

konštanta v BCM 2835	konštanta v aplikácii Gpio
RPI_GPIO_P1_18	5
RPI_GPIO_P1_11	0
RPI_GPIO_P1_12	1
RPI_GPIO_P1_15	3

Tabuľka 3: Rozdiel medzi označením pinov v aplikácii *Gpio* a knižnicou BCM 2835

Viac o projekte Wiring a jeho aplikáciach je možné vidieť na jeho web stránke [15].

6.2 Implementačný jazyk a použité nástroje

Na implementáciu som si zvolil jazyk C, ktorý patrí medzi kompilované jazyky a dá sa obohatiť o množstvo prídavných funkcií. Kompilácia prebieha prekladačom (v systéme Linux je to Gcc), ktorý zo zdrojového kódu napísaného v jazyku C, vygeneruje spustiteľný súbor.

Ako výhody tohto jazyka uvádzam:

- vysoká pamäťová efektivita programov a výpočtová rýchlosť
- možnosť nízkoúrovňových operácií pre priamu komunikáciu s hardvérom.
- široká dostupnosť kvalitných prekladačov a knižníc.

Ďalej uvádzam, aké nástroje boli potrebné pri vývoji tohto softvéru

- Linux Ubuntu 14.04 32bit

[15] `http://wiringpi.com/`

- Code::Blocks 12.1
- GNOME Terminal 3.4.1.1
- Kompilátor gcc
- Make
- Verzovací systém git
- Logovací systém Syslog
- Knižnica NCURSES 5.9
- Knižnica I2C-dev
- Knižnica PThread
- Knižnica NMEA
- Knižnica BCM 2835
- Knižnica libDVB

Dokumentácia programu SatBerry a jeho teória je napísaná v typografickom systéme LaTeX. Jednotlivé diagramy sú nakreslené za pomoci nástroja Violet UML Editor, ktorý je šíriteľný pod voľnou GPL licenciou.

Tak, ako uvádza zdroj [19], použitie logovacieho systému **Syslog** je dôležité pre analýzu chybovosti rôznych programov fungujúcich pod Linuxom. Po pridaní hlavičky `#include <syslog.h>` do zdrojového kódu je možné využiť záznamovú funkciu `syslog()`, ktorej vstupné argumenty sa ukladajú priamo do systémového logu *syslog* uloženého v adresári /var/log/. Pred prvým zápisom je potrebne log otvoriť a určiť formát zápisu. O to sa stará funkcia `openlog()`. Podľa úrovne záznamu sa určí priorita a ciel zápisu. V aplikácií boli použité len dve úrovne logovania:

1. `LOG_ERR` - úroveň pre chybové výstupy. Zapisuje sa do `/var/log/syslog`
2. `LOG_NOTICE` - základná informačná úroveň. Zapisuje sa do `/var/log/messages`

Nakoľko mobilná satelitná anténa nemá štandardne obrazový výstup, kde by bolo možné diagnostikovať poruchy, bol som nútený logovať kritické operácie do súborového systému v systémovom logu pre prípadnú diagnostiku skúsenejším užívateľom. Spomenuté logovanie som využil v najdôležitejších funkciách pri načítavaní hodnôt z čidiel.

6.3 Použité prídavné knižnice

Na účel mojej aplikácie samozrejme nestačil štandardný balík funkcií jazyka C, ale bol som nútený siahnuť po špecifickejších balíkoch, zaoberajúcich sa požadovanou tématikou, nazývané knižnice (angl. libraries), ktoré mali v sebe zabalené potrebné funkcie.

1. NCURSES
2. I2C-dev
3. NMEA
4. Pthread
5. BCM 2835
6. libDVB
7. libmath-dev

Všetky knižnice, až na NMEA a BCM, je možné v Linux systéme Raspbian jednoducho nainštalovať cez aplikačný program `apt-get`, nakoľko použitý OS Raspbian je derivát Debianu, ako aj Ubuntu, a preto sa nachádzajú v ich repozitároch.

Aby kompilátor vedel, že má možnosť použitia väčšieho rozsahu funkcií obsiahnutých v knižniciach, tak mu je ich potrebné nalinkovať a pridať ich hlavičky do zdrojového kódu. Knižnice sa linkujú s parametrom -l a hneď za ním názov knižnice. Tento parameter sa prekladaču zadáva ako posledný. Napríklad prekladaču Gcc, ktorý kompiluje súbor main, dávame k dispozícií knižnicu, tak jeho zápis bude vyzerať nasledovne:

```
gcc main.c –o main –lkniznica
```

6.3.1 Knižnica NCURSES

Názov knižnice Ncurses je zložený zo slov new curses. Ide o voľnú softvérovú emuláciu curses System V Release 4.0. Používa terminálový formát, podporuje odsadenia, farby, zvýrazňovanie, formuláre a mapovanie funkčných kláves. Obsahuje radu vylepšení oproti BSD curses. Ncurses kód bol vyvinutý v rámci GNU / Linux. Používala sa nejakú dobu s OpenBSD ako systémová Curses knižnica na FreeBSD a NetBSD ako externý balík. Mala by byť kompatibilná s akoukoľvek distribúciou UNIXu typu ANSI / POSIX a tiež dokonca s OS / 2 Warp.

Pri implementácií projektu mali tieto funkcie najväčší výskyt a najväčšie opodstatnenie:

- `initscr(), endwin()` - štart/stop grafického módu Ncurses.
- `newwin()` - vracia ukazovateľ na obrazové okno.
- `wrefresh()` - vykresľuje okno na obrazovku.

- `printw(), mvprintw(), mvwprintw(), mvaddstr()` - výsadba textu na obrazovku, podľa zadaných parametrov.
- `attron(), attroff()` - riadenie formátovania znakov.

Do systému bol nainštalovaná cez klasický inštalačný program, nakoľko je dostupná z repozitárov, zadaním príkazu: `apt-get install libncurses5-dev`
Zdroj: [20]

6.3.2 Knižnica I2C-dev

Knižnica sa nám stará o komunikáciu medzi I2C zariadeniami za pomoci SMBus (angl. System Management Bus) protokolu, ktorý tvorí podmnožinu I2C protokolu. Príkazy pre zbernicu SMB sú automaticky konvertované pre I2C zbernicu a opačným spôsobom to nieje vždy možné.

Ako bol spomenuté v úvode podkapitoly 6.3, knižnica je dostupná z Linux repozitárov. Jej inštalácia prebehla zadaním príkazu: `apt-get install libi2c-dev`

Komunikácia s I2C rozhraním prebiehala v niekoľkých krokoch a za pomoci nasledovných funkcií:

1. `open()`
2. `ioctl()`
3. - `i2c_smbus_read_word_data()`
   - `i2c_smbus_read_i2c_block_data()`
4. `close()`

V prvom kroku je nutné dané I2C rozhranie otvoriť systémovou funkciou `open()`, ktorá priradí deskriptor I2C rozhrania k premennej súboru, s ktorým sa už ďalej pracuje. V mojom prípade jej použitie vyzerá nasledovne: `file = open(portName, O_RDWR)`

Prvý argument je textová premenná, ktorá predstavuje absolútnu linuxovú adresu I2C rozhrania, v mojom prípade to je `/dev/i2c-1` a druhý argument je konštanta, ktorá určuje spôsob komunikácie, v mojom prípade obojsmernú komunikáciu (čítanie aj zápis).

V druhom kroku, po otvorení rozhrania, je možné cez neho poslať HEX adresu konkrétneho I2C zariadenia, s ktorým sa bude komunikovať. Na tento účel slúži nízkoúrovňová funkcia `ioctl()` ktorej parametrami je súborová premenná, konštanta podľa typu požiadavky a prenášaná hodnota. V mojom prípade išlo o komunikáciu s I2C slave zariadením, ktoré bolo potrebné aktivovať na konkrétnej I2C slave adrese.

V mojom prípade jej použitie vyzerá nasledovne: `ioctl(file, I2C_SLAVE, address)`

V treťom kroku sa už používajú čítacie funkcie obsiahnuté v I2C-dev knižnici. Je ich niekoľko typov a delia sa podľa veľkosti (dátových typov) prenášaných dát, boli však použité len dve.

V prvom prípade chcem vrátiť 16 bitovú WORD hodnotu a jej argumentami je súborová premenná I2C rozhrania a HEX adresa pamäte z ktorej vyťahujem hodnotu. WORD hodnota sa skladá z

- **LSB** - spodný bajt, tvorí prvých 8 bitov z prava.
- **MSB** - horný bajt, tvorí prvých 8 bitov z ľava.

V mojom prípade jej použitie vyzerá nasledovne: `i2c_smbus_read_word_data (file, daddress)`

V druhom prípade chcem načítať celý blok dát, ktorých veľkosť je určená tretím parametrom a do štvrtého parametru sa celý blok ukladá. Prvé dva funkčné parametre majú rovnaký účel, ako v predchádzajúcom prípade.

V mojom prípade jej použitie vyzerá nasledovne:

```
i2c_smbus_read_i2c_block_data(file, daddress, BMA180_BUFFER,
accVar->dataBuffer)
```

Vo štvrtom kroku sa končí komunikácia a uzatváram I2C rozhranie funkciou, ktorá má jediný parameter a to súborovú premennú. V mojom prípade jej použitie vyzerá nasledovne: `close(file)`.

Zdroj: [21]

6.3.3 Knižnica Pthread

Táto POSIX knižnica je založená na API štandarde vlákien pre jazyk C/C++. To umožňuje, aby jeden proces vytváral súbežné nové procesné toky. Najviac je to efektívne s viacerými procesormi alebo s viacjadrovými systémami, kde prevádzka procesného toku môže byť rozvrhnutá na inom procesore a tým získava rýchlosť cez paralelný beh alebo distribuované spracovanie.

Vlákna vyžadujú menšie nároky na vytvorenie nového procesu, vetvenia, pretože systém neinicializuje novú virtuálnu pamäť a prostredie. Všetky vlákna v rámci procesu zdieľajú rovnaký adresový priestor. Vláknu treba definovať funkciu a jej argumenty, ktoré sa budú v nej spracovávať.

Účelom použitia vlákien POSIX v projekte SatBerry je neustále čítanie hodnôt z I2C senzorov pri diagnostike (v manuálnom režime). Každé vlákno sa staralo o čítanie a zobrazovanie jedného senzora. Vďaka tomu je možné na obrazovke vidieť všetky aktuálne údaje naraz.

Avšak spôsob pripojenia k GPS senzoru a DVB-S karty neumožňoval viacnásobný prístup, a preto sú tieto dve vlákna štandardne vypnuté, aby mohli byť tieto senzory využité v hlavnom vlákne, v ktorom podla nich prebieha smerovanie.

Pri programovaní utility SatBerry boli z tejto knižnice najčastejšie použité tieto funkcie:

- `thread_create()`, `pthread_cancel()`- spúšťa, ukončuje zadefinované vlákno s parametrom.
- `pthread_join()`- čaká na dokončenie vykonávania zadefinovaného vlákna.

- `pthread_mutex_lock()` - pozastavenie vykonávania zadefinovaného vlákna.

Pri neustálom čítaní hodnôt z I2C zariadení bolo potrebné nastaviť časový rozostup medzi cyklusmi (napríklad hodnota sa za stotinu sekundy nezmenila), čiže zabezpečiť chvíľkové uspanie vlákna. V tejto funkcii som musel použiť takzvaný semafor, kde pozastavím beh vlákna na určitú dobu. Podrobnejšie v nižšie uvedenom výpise 1.

```
void waiting(int seconds)
{
    struct timeval now;
    struct timespec WaitingTime;

    gettimeofday(&now,NULL);
    WaitingTime.tv_sec = now.tv_sec + seconds;
    WaitingTime.tv_nsec = now.tv_usec;

    pthread_mutex_lock(&threadMutex);
    pthread_cond_timedwait(&threadCondition, &threadMutex, &WaitingTime);
    pthread_mutex_unlock(&threadMutex);
}
```

Výpis 1: Funkcia na pozastavenie behu vlákna

Vo výpise 1 je ukázané použitie zámku `pthread_mutex_t threadMutex` a vláknovej podmienky `pthread_cond_t threadCondition`. Pomocou zámku procesor pozastaví vlákno a spustí ho až po splnení podmienky, v mojom prípade časovej. Funkciu `waiting()` som si vytvoril pre vytvorenie časového rozostupu pri neustálych aktualizáciách dát.

Zdroj: [22]

6.3.4 Knižnica NMEA

Predstavuje open source voľnú knižnicu, napísanú v programovacom jazyku C, pre prácu s NMEA protokolom. Medzi jej výhody patria:

1. analýza a spracovanie GPS dát do C štruktúr.
2. generovanie NMEA viet.
3. viacvrstvová architektúra a algoritmy.
4. multiplatformosť (UNIX, Windows, Windows Mobile) .
5. podpora viet: GPGGA, GPGSA, GPGSV, GPRMC, GPVTG.

Programová časť súvisiaca s GPS implementáciou, bola prevzatá od vedúceho tejto práce. Konkrétne sa jedná o funkcie:

- `initializeGps()`
- `closeGps()`

- `gpsIsInitialized()`
- `getGpsInfo()`
- `nmea_ndeg2degree()`

Táto časť kódu je zadefinovaná v súboroch *gps.c* a *gps.h*. Ide predovšetkým o otvorenie sériového rozhrania */dev/ttyAMA0* a jeho nastavenie podľa NMEA štandardu pre účel komunikácie spomenutej v podkapitole 4.2.1. Následne vo zvolenej frekvencií prebieha čítanie dát (viet) a ich transformovanie NMEA funkciou `nmea_parse()` do pripravenej NMEA štruktúry.

Na základe údajov z GPS senzora, algoritmov vo funkciách `getElevationPos()` a `getAzimuthPos()`, viem vypočítať potrebný uhol na smerovanie paraboly k družici, ako bolo spomenuté v kapitole 2.2. Tieto výsledky, spolu s údajmi o polohe, som uložil do mojej zmysluplnej informačnej štruktúry `position`, ktorú napĺňam vo funkcií `getPosition()`, kde prebieha volanie vyššie spomenutých operácií. Danú štruktúru som potom mohol využívať na nastavenie potrebnej elevácie a azimutu v polohovej časti programu.

Zdroj: [23]

6.3.5 Knižnica BCM 2835

Jednou z najdôležitejších knižníc je práve knižnica BCM 2835, určená na správu rovnomenného SoC čipu od spoločnosti Broadcom. Poskytuje prístup k GPIO rozhraniu a k ďalším IO funkciám Broadcom BCM 2835 čipu.

Umožňuje ovládať vstupy aj výstupy na dvadsiatich-šiestich pinoch GPIO na doske RPi. Cez toto rozhranie je možné ovládať rôzne externé zariadenia. Knižnica obsahuje funkcie na čítanie digitálnych vstupov, nastavovanie digitálnych výstupov pomocou SPI a I2C a tiež funkcie na prístup k systémovému časovaču. Udalosti vzniknuté prerušením obvodu na pinoch nie sú podporované.

SoC BCM 2835 používa rovnaké číslovanie pinov GPIO podľa fyzického zapojenia za sebou a označenia na doske RPi. Avšak nie všetky piny sú dostupné prostredníctvom tejto knižnice. V nasledujúcej tabuľke 4 je ukázané, ktoré piny a ich konštanty boli použité v programe SatBerry pre aktiváciu elektromotorčeka želaným smerom.

Smer	konštanta v BCM 2835	fyzické značenie na RPi v.1
RIGHT	`RPI_GPIO_P1_18`	Pin P1-18
LEFT	`RPI_GPIO_P1_11`	Pin P1-11
UP	`RPI_GPIO_P1_12`	Pin P1-12
DOWN	`RPI_GPIO_P1_15`	Pin P1-15

Tabuľka 4: Mapovanie GPIO pinov

Knižnica je kompatibilná s programovacím jazykom C++ a je ju možné nainštalovať na akúkoľvek distribúciu na báze Linuxu. Nasledovný výpis ukazuje, ako ju stiahnuť a skompilovať.

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.38.tar.gz
tar zxvf bcm2835-1.14.tar.gz
cd bcm2835-1.xx
./configure
make
sudo make check
sudo make install
```

Zdrojové kódy sú uvoľnené pod open source licenciou a vývoj pokračuje. Pracovalo sa s verziou 1.14 a v čase písania záverečnej práce bola už k dispozícií verzia 1.38, kde sa testuje kompatibilita s najnovšou verziou RPi v.2.

Vystačil som si s nasledovnými funkciami:

- `bcm2835_gpio_fsel(GPIOportID, BCM2835_GPIO_FSEL_OUTP)` - inicializuje daný GPIO port ako výstupné rozhranie.
- `bcm2835_gpio_write(GPIOportID, HIGH)` - na danom GPIO porte sa zapne napätie a v prípade parametru LOW sa vypne.
- `bcm2835_init(), bcm2835_close()` - štart / stop komunikácie s GPIO rozhraním.

Vyššie uvedené funkcie sú plne postačujúce na ovládanie pohybu antény. Pre dosiahnutie relevantného výsledku pri používaní elektrického motorčeka sa vždy na určitý čas privedie napätie na výstup GPIO pinu a po jeho uplynutí vypne. Vzhľadom na to, že nebol k dispozícií krokový motorček na smerovanie paraboly, motorická logika bola nastavená tak, aby vždy po krátkom pohybe overil systém svoju pozíciu a pokračoval ďalej v pohybe.

Zdroj: [24]

6.4 Kompilácia a krížená kompilácia

Aby som zrýchlil kompiláciu a nemusel využívať slabší hardvér RPi, nainštaloval som nástroj na kríženú (angl. cross) kompiláciu **arm-linux-gnueabi-gcc** na výkonnejšiu pracovnú stanicu s procesorom Intel i7 s architektúrou i386, kde prebiehal vývoj. Pri použití tohto kompilátora sa program neprekladá do architektúry i386 (kde je spúšťaný), ale do ARM. Tento takzvaný krížený kompilátor je dostupný v repozitároch Debianu (a v jeho derivátoch) a je k dispozícií vo viacerých formách podľa cieľového kompilovaného programovacieho jazyka.

Inštaluje sa zadaním príkazu: `apt-get install gcc-arm-linux-gnueabi`

Ešte pred kríženou kompiláciou vyvíjaného programu je potrebné krížením skompilovať aj vložené knižnice. Krížová kompilácia **knižnice BCM 2835** sa vykonáva nasledovným spôsobom:

```
CC=arm-linux-gnueabi-gcc
./configure arm-linux --host=arm-linux-gnueabi
```

```
make
make install
```

Pre kríženú kompiláciu **knižnice NCURSES** je to takmer rovnaké, ale je potrebné mať nainštalovaný krížený kompilátor pre jazyk C++ a ešte zadefinovať:

```
make HOSTCC=gcc CXX=arm-linux-gnueabi-g++
```

Krížová kompilácia **knižnice NMEA** je mierne odlišná, pretože pre ňu je potrebné zmeniť typ kompilátoru v súbore *Makefile* z `gcc` na `arm-linux-gnueabi-gcc` a až potom spustiť inštaláciu príkazom `make all`.

Knižnice, ktoré nie sú natívnou súčasťou OS na pracovnej stanici boli krížením skompilované do *a archívu* a uložené do zložky `./libCross`. Išlo o knižnice NMEA a BCM 2835.

Problém však nastal pri kríženej kompilácii knižnici NMEA, kde **GPS senzor vracal nulové hodnoty** bez akejkoľvek chyby. Rovnaké správanie sa ukázalo aj pri použití inej GPS knižnice (libgps)[16]. Správne fungovanie GPS knižnice zabezpečila až **zmena z dynamického linkovania na statické** (prepínač kompilátora -s), ktoré však zväčšovalo veľkosť programu, keďže tam pridávalo všetky knižnice. Z toho dôvodu som bol nútený s kríženou kompiláciou skončiť a prejsť na vzdialené volanie programu `make` a kompilovať na diaľku lokálne, kde knižnica NMEA fungovala korektne.

Implementačné zmeny prebiehali na súborovom systéme RPi pripojeného prostredníctvom protokolu SFTP. Vtedy sa vzdialený sieťový priečinok tvári ako lokálna zložka.

6.5 Spôsoby implementácie niektorých funkcií

Smerovanie paraboly sa nezaobíde bez funkcií na zmenu azimutu a elevácie. Implementoval som rôzne pohybové funkcie do projektovej časti *positioning*, kde je riešená aj aktivácia GPIO rozhrania, pretože zmena pozície s ňou súvisí. Jedná sa predovšetkým a nie výlučne o nasledovné pohybové funkcie:

- **jednoduché** - pohyb podľa smeru (`moveLEFT()`, `moveRIGHT()`, `moveUP()`, `moveDOWN()`).
- **základné** - čítanie „get funkcie“, nastavenie „set funkcie“ azimutu a elevácie podľa skutočnej hodnoty senzora (`setAzimuthCompass()`, `getElevationPos()`).
- **komplexné** - využívajú kombináciu predchádzajúcich a dopočítavajú ďalšie hodnoty (`moveAzim()`, `moveElev()`).

Prakticky, ako vyzerá funkcia `moveAzim()` je ukázané v nasledovnom výpise 2, pričom na rovnakom princípe je naprogramovaná funkcia `moveElev()`.

```
void moveAzim(int grades, char direction)
{
    int angleAct = getAzimuthRelative();
    if (direction == 'R')
```

[16] `https://github.com/wdalmut/libgps`

```
            setAzimuthRelative(angleAct − grades);
        else if (direction == 'L')
            setAzimuthRelative(angleAct + grades);
        else
            return;
    }
```

Výpis 2: Smerovacia funkcia azimutu podľa smeru a počtu stupňov

Vyššie uvedená funkcia vo výpise 2 dostáva parametre na nastavenie smeru otáčania, čo môže byť doprava **R**, alebo doľava **L** a počet stupňov, o koľko sa má otočiť. Najprv si zistí v akej relatívnej pozícií sa nachádza funkciou `getAzimuthRelative()`, urobí prepočet uhla na ktorý sa má nastaviť a to hlavné posunutie už rieši funkcia `setAzimuthRelative()`, ktorá riadi pohybový mechanizmus a pri každej zmene zisťuje, či sa už v požadovanom uhle nenachádza.

6.5.1 Jemné ladenie

Po nasmerovaní antény podľa GPS súradníc na daný azimut a eleváciu je ešte potrebné jemne doladiť (pohýbať), aby sa dosiahol čo najlepší DVB-S signál. Na jemné dolaďovanie bola implementovaná funkcia `tuneSignalFine()`, ktorá očakáva jediný vstupný parameter a to ukazovateľ na súborové rozhranie DVB-S karty, ktoré bolo predtým inicializované. Jej výstupom je hodnota najlepšieho DVB-S signálu, ktorá sa potom zobrazuje.

Pre rozsiahlosť funkcie `tuneSignalFine()` len v bodoch zhrniem, ako daná funkcia funguje a v akom poradí vykonáva jednotlivé operácie.

1. definuj najlepší DVB-S signál = 0.
2. definuj najlepší smer, ako ten v ktorom sa momentálne nachádzaš `middleMiddle`.
3. zmeraj DVB-S signál.
4. posun sa o stupeň dole.
5. zmeraj DVB-S signál.
6. ak je signál lepší (väčší), ulož jeho hodnotu a tiež smer v ktorom si.
7. posun sa na ďalší smer o stupeň doľava (pokračuje sa v smere hodinových ručičiek)
8. opakuj akciu z bodu 5., 6. a 7..
9. po prejdení všetkých bodov sa vráť na počiatočný stred (smer `middleMiddle`).
10. zavolaj funkciu `setDirection()`, na nastavenie najlepšieho smeru s najlepším signálom, ak to nieje tá, v ktorej sa nachádzaš (`middleMiddle`).

Ide v podstate len o zmenu polohy o jeden stupeň všetkými smermi, ktoré tvoria 9 možností. Tieto možnosti som si definoval dátovým typom `enum`. Po ich prejdení sa nastaví anténa na najlepší z nich.

6.6 Ladenie a testovanie

Pre funkčný systém ladenia v aplikácii SatBerry a aktivácie prijímania DVB-S signálu je potrebné priviesť napätie cez koaxiálny kábel až k LNB konvertoru. Obvykle toto napätie dodáva prijímajúci set-top box, ktorý si ním aj riadi polarizáciu antény, ale je ho možné nastaviť aj na výstupe USB DVB-S karty, ktorá načítava hodnoty DVB-S signálov. V mojom prípade napätie k LNB dodáva set-top box a pre ovládací kontrolér, spolu s motorčekmi, dodáva napätie adaptér s výstupom 7,5V pri jednosmernom prúde 1A.

Po pripojení potrebných elektrických zdrojov a naštartovaní operačného systému RPi sa vyvíjaná utilita spustí sama v automatickom režime. Môže však pracovať v dvoch režimoch:

1. **automatický** - prebieha nekonečný cyklus hľadania signálu z GPS, DVB-S a následne polohovanie paraboly a jej jemné ladenie.
2. **manuálny** - diagnostický, kde jednotlivé funkčné časti sú spúšťané manuálne.

Pri manuálnom režime je spúšťaná v tvare `.\sattberry -m`, kde je zobrazený konzolový obrazový výstup a k dispozícií možnosti spomenuté v návrhovej časti tejto práce, v kapitole 5.2. Ovládanie, ktoré nevyplýva z grafického rozhrania programu je nasledovné:

- **malý pohyb** - podľa smerových klávesových šipiek.
- **zmena polohy paraboly o 5 stupňov** - klávesami „w“, „s“, „a“, „d“.
- **naladenie signálu** - klávesou „z“.
- **jemné doladťovanie** - klávesou „x“.

Na testovanie digitálneho DVB-S video signálu boli potrebné aplikácie a utility dostupné z LINUX repozitárov. Boli použité programy z balíčkov dvb-apps a w-scan.

6.6.1 Softvérom tretích strán

Pred samotným spustením aplikácie SatBerry som si otestoval prítomnosť satelitného hardvéru. Nasledovné utility zobrazujú aj presnejšie špecifikácie pripojeného hardvéru:

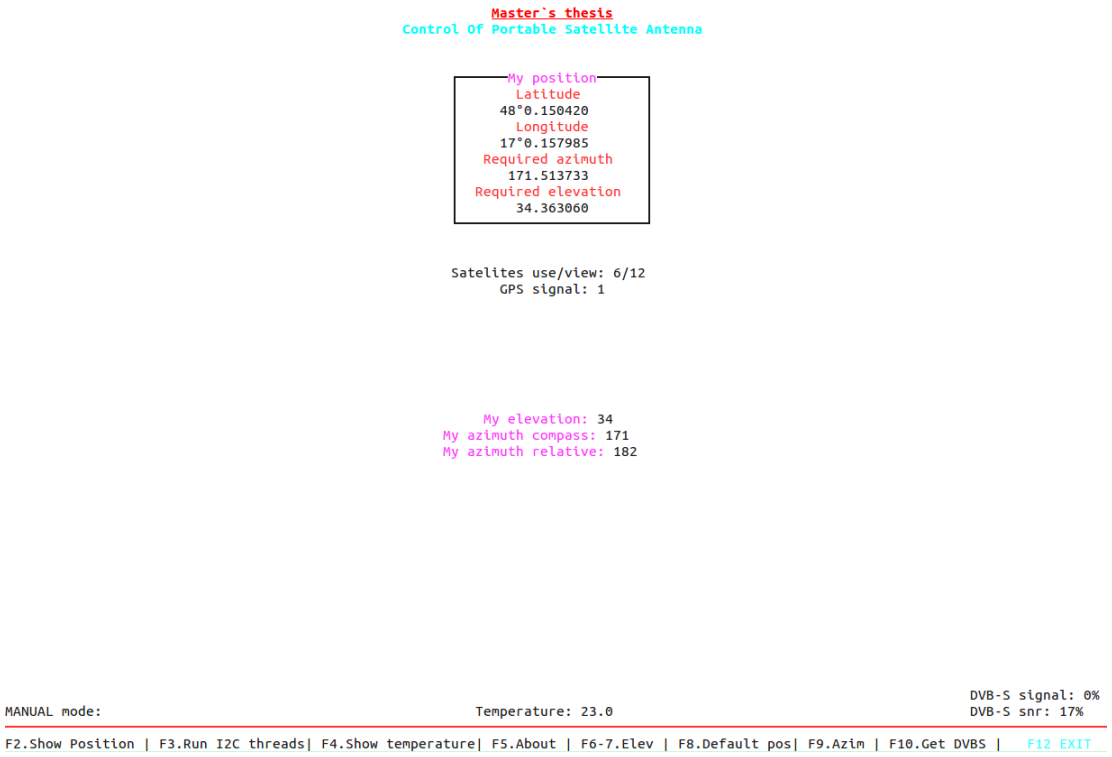
- **lsdvb** - testovanie prítomnosti DVB zariadenia
- **lsusb** - testovanie prítomnosti DVB zariadenia pripojeného cez USB rozhranie, systémová utilita.
- **dvbsnoop** -adapter 0 -s feinfo - modulačné informácie o DVB zariadení
- **dmesg | grep -i dvb** - kontrola inicializácie DVB tuneru

Testovanie signálu som potom vykonával nasledovnými pomocnými nástrojmi a ich parametrami:

- **dvbsnoop** -n 3 -pd 3 -s signal
- **femon** -H
- **scan** -c
- **w_scan** -f s -c SK -s S23E5

6.6.2 Vyvíjanou utilitou SatBerry

Po spustení utility je zobrazené nasledovné grafické rozhranie.



Obr. 13: Grafické rozhranie SatBerry (zdroj vlastný)

Ako bolo spomenuté v podkapitole 6.6, tak s prepínačom `-m` sa spustí diagnostické grafické rozhranie programu v manuálnom režime, kde možno s funkčnými klávesami **F2-F10, F12** spustiť jednotlivé operácie. V strede obrazovky je najdôležitejšie okno s údajmi o polohe a s potrebnými smerovacími uhlami. Hneď pod GPS oknom sa nachádzajú informácie o sile GPS signálu a počtu viditeľných GPS satelitov. V pravo dole sa ukazuje hodnota DVB-S signálu a jeho SNR rušenia. V ľavo dole je zobrazovaný práve spustený režim aplikácie.

DVB-S signál sa nepodarilo namerať ani v jednom z použitých nástrojoch a ani vyvíjanou utilitou SatBerry. Nieje to chyba implementácie a ani konfigurácie, ale pravdepodobne hardvérovou chybou na LNB žiariči, nakoľko signál bol prítomný v počiatočnom

štádiu vývoja, keď som testoval ešte len pomocou softvéru tretích strán. Jediným funkčným ukazovateľom sily signálu bola veľkosť jeho rušenia tzv. signal-to-noise ratio (v skratke SNR).

Podla hodnôt SNR som bol schopný simulovať algoritmus jemného ladenia avšak pri opačnej logike, kde najlepší výsledok nedosiahneme najvyššou hodnotou signálu, ale najmenšou hodnotou rušenia. Nepresné výsledky algoritmu spôsobili veľmi nestabilné hodnoty SNR aj počas nemennej polohy, a preto sa musím spoliehať len na naladenie signálu podľa GPS polohy.

6.7 Ďalšia práca

Ďalšia práca by mohla viesť k zlepšeniu rýchlosti programu. V implementácii som použil pohotovostný mód pre senzor kompasu a ten by sa dal prepnúť do vyhľadávajúceho módu a až potom z neho čítať dáta, čím by sa urýchlil proces čítania dát z kompasu.

Spôsob komunikácie s I2C rozhraním by sa mohol implementovať ako u DVB-S komunikácie. Čiže rozdeliť funkciu na získanie hodnoty I2C na tri časti:

1. otvorenie rozhrania.
2. čítanie z rozhrania v cykle, alebo na požiadanie do ukončenia programu.
3. zatvorenie rozhrania.

Taktiež by bolo vhodné optimalizovať algoritmy zapisujúce do logov, ktoré sú uložené na disku, pretože je potrebné minimalizovať zápisy na SD kartu, nakoľko je limitovaná počtom zápisov a takto znižujeme jej životnosť.

7 Záver

V tejto práci sa analyzovali geostacionárne satelity, ich distribúcia signálu a opísaná bola ich konštrukcia. Na základe poznatkov z ich teórie bolo možné skonštruovať mobilnú satelitnú anténu, avšak túto časť práce za mňa vykonal vedúci diplomovej práce a mojou úlohou bolo implementovať do nej programovú logiku.

Pri návrhu konštrukcie mobilnej satelitnej antény som zistil cenu originálneho výrobku, ktorá sa na trhu pohybuje v rozmedzí 600 - 1000 $ (podľa hardvérového vybavenia). Teraz po jej zhotovení viem odhadnúť, že finančný náklad celého riešenia mobilnej satelitnej antény vyšiel približne na 554 $.

Musel som sa naučiť spracovávať dáta z rôznych polohových, či satelitných senzorov a zvládol som komunikovať s dvoj-vodičovou I2C zbernicou, ktorá sa bežne používa v integrovaných obvodoch, ako to bolo v tomto prípade medzi polohovými senzormi. Medzi ne patrí napríklad použitý kompas, akcelerometer a A/D prevodník. Boli vysvetlené smerovacie informácie, ktoré sa získavajú z GPS senzora v podobe NMEA viet a jednu bola ukázaná na príklade.

Zaviedol som rôzne polohy relatívneho azimutu voči jeho absolútnej reálnej hodnote. Oboje bolo nutné rozlišovať v smerovacích algoritmoch paraboly a ukázané na príkladoch.

Po správnom natočení paraboly, vzhľadom k polohe GPS, som implementoval jemné ladenie, ktoré sa prispôsobovalo najlepšiemu získanému DVB-S signálu. Avšak tu som narazil na problém s mechanickým poškodením LNB a nebol som schopný načítať hodnotu DVB-S signálu s naprogramovanou aplikáciou a ani s jedným podporným softvérom. Ani priamo pripojený set-top box nevedel získať DVB-S signál z LNB konvertoru. Na základe týchto faktorov konštatujem, že pri vývoji došlo k poškodeniu LNB.

Anténa sa ladí len podľa získaných GPS súradníc a ich prepočte na správny smerovací uhol azimutu a elevácie. Kvalitu aplikácie to veľmi neovplyvnilo, lebo všetky potrebné funkcie boli implementované a dá sa predpokladať, že po výmene poškodeného LNB konvertora bude mobilná satelitná anténa plne funkčná.

Došiel som k myšlienke, že nestrávim ani jednu jedinú chvíľu na cestách, na dovolenke bez svojho obľúbeného televízneho seriálu. Dopomôže mi k tomu práve taká malá praktická mobilná satelitná anténa a je to jedna z prvých vecí, ktorú si zbalím.

K výsledku by som sa nedopracoval, keby nebolo malého a všestranného mikro počítača Raspberry Pi. Jeho popularita stále naberá na obrátkach. Veď od uvedenia prvého modelu v roku 2012 predali už 5 miliónov kusov a nový, 6 krát výkonnejší model RPi 2, bol dostupný na trhu len 3 týždne a už sa ho za tú dobu predalo viac ako pol milióna kusov.

Ľubomír Wenzl

8 Literatúra

[1] REHÁK, Dušan. *Satelitná komunikácia a služby umožňujúce mobilitu v TCP/IP sieťach*. Brno, 2004. Diplomová práca, Fakulta informatiky MU Brno. Tiež dostupné z: <http://www.fi.muni.cz/usr/staudek/rehak/diplomovka.html>

[2] DÁVIDIK, Tomáš. *Satelity » Teória pohybu satelitov - Hardware.sk [online]*, last revision 23.3.2007 [vid. 9.9.2014]. Dostupné z: <http://www.hardware.sk/clanok-577/teoria-pohybu-satelitov-/>

[3] SATBEAMS. *SatBeams - Satellite Details - Astra 3B [online]*, last revision 2014 [vid. 19.9.2014]. Dostupné z: <http://www.satbeams.com/satellites?norad=36581>

[4] NODŽÁKOVÁ, Zuzana, Ing. *LNB konvertory - Satelitná technika - Ellano.sk [online]*, last revision 2015 [vid. 12.9.2014]. Dostupné z: <http://www.satelity.ellano.sk/lnb-konvertory_203.html>

[5] MARCHEVSKÝ, Stanislav. PILLAR, Slavomír. HRUŠOVSKÝ, Branislav. *Satelitné technológie a služby*. Košice, 2011. Skripta, Fakulta elektrotechniky a informatiky TU Košice. s.77-82. Tiež dostupné z: <http://www.kemt-old.fei.tuke.sk/predmety/KEMT559_SK/_materialy/materialy_macekova/STS_ext/MarchPillHrus_STS_v15_17_5_2011.pdf>

[6] SPARKFUN Electronics. *Compass Module - HMC6352 - SEN-07915 - SparkFun Electronics [online]*, last revision 2015 [vid. 11.10.2014]. Dostupné z: <https://www.sparkfun.com/products/retired/7915/>

[7] SPARKFUN Electronics. *Triple Axis Accelerometer Breakout - BMA180 - SEN-09723 - SparkFun Electronics [online]*, last revision 2015 [vid. 11.10.2014]. Dostupné z: <https://www.sparkfun.com/products/retired/9723/>

[8] MICROCHIP Technology Inc. *MCP3221 - Mixed Signal- Successive Approximation Register (SAR) A/D Converters [online]*, last revision 2014 [vid. 12.10.2014]. Dostupné z: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010535&redirects=mcp3221>

[9] SPARKFUN Electronics. *GPS Receiver - LS20031 5Hz (66 Channel) - GPS-08975 - SparkFun Electronics [online]*, last revision 2015 [vid. 12.10.2014]. Dostupné z: <https://www.sparkfun.com/products/8975/>

[10] CITYCOM GmbH. *TT-connect® S-2400 [online]*, last revision 2015 [vid. 13.10.2014]. Dostupné z: <http://www.technotrend.eu/2778/TT-connect__S-2400.html>

[11] UPTON, Eben a Gareth HALFACREE. *Raspberry Pi user guide.* 2. aktualiz. vyd. Chichester, England: John Wiley, c2012. xiv. ISBN 11-184-6446-X.

[12] TRAYNOR, Bill. *RPi Hardware - eLinux.org [online]*, last revision 28.2.2015 [vid. 1.3.2015]. Dostupné z: <http://elinux.org/RPi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29>

[13] DIDH Network L.L.C. *DISH Network Tailgater - USDish [online]*, last revision 2015 [vid. 16.10.2014]. Dostupné z: <http://www.usdish.com/equipment/tailgater/>

[14] HW server s.r.o. *Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877 | HW.cz [online]*, last revision 2014 [vid. 20.9.2014]. Dostupné z: <http://www.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256/>

[15] NXP Semiconductors N.V. *I2C-bus specification and user manual [online]*, last revision 6.4.2014 [vid. 2.9.2014]. Dostupné z: <http://www.nxp.com/documents/user_manual/UM10204.pdf>

[16] DEPRIEST, Dale. *NMEA data [online]*, last revision 2008 [vid. 10.9.2014]. Dostupné z: <http://www.gpsinformation.org/dale/nmea.htm>

[17] EDGEWALL Software. *i2cToolsDocumentation – lm-sensors [online]*, last revision 2009 [vid. 10.10.2014]. Dostupné z: <http://www.lm-sensors.org/wiki/i2cToolsDocumentation/>

[18] HENDERSON, Gordon. *Raspberry Pi | Wiring | Download & Install | Wiring Pi [online]*, last revision 2015 [vid. 28.9.2014]. Dostupné z: <http://wiringpi.com/download-and-install/>

[19] SKOČOVSKÝ, Luděk a Scott JERNIGAN. *Linux: dokumentační projekt.* 4. aktualiz. vyd. Překlad Ivo Fořt, David Krásenský. Brno: Computer Press, 2007. 1334 s. ISBN 978-80-251-1525-1.

[20] DICKEY, Thomas. *Announcing ncurses 5.9 - GNU Project - Free Software Foundation (FSF) [online]*, last revision 4.4.2011 [vid. 28.02.2013]. Dostupné z: <http://www.gnu.org/software/ncurses/ncurses.html>

[21] LINUX, Kernel Organization, Inc. *Documentation of I2C SMBus protocol [online]*, last revision 3.2.2015 [vid. 5.2.2015]. Dostupné z: <https://www.kernel.org/doc/Documentation/i2c/smbus-protocol/>

[22] IPPOLITO, Greg. *Linux Tutorial: POSIX Threads [online]*, last revision 2014 [vid. 10.11.2014]. Dostupné z: <http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html>

[23] SPACEFLUSH, Tim. *NMEA library [online]*, last revision 2014 [vid. 10.10.2014]. Dostupné z: <http://nmea.sourceforge.net/#features>

[24] MCCAULEY, Mike. *C library for Broadcom BCM 2835 as used in Raspberry Pi [online]*, last revision 2015 [vid. 10.02.2015]. Dostupné z: <http://www.airspayce.com/mikem/bcm2835/>

A Elektronická príloha

Disk DVD obsahuje okrem PDF verzie tohto dokumentu aj zdrojové kódy k Satberry aplikácií. Taktiež obsahuje firmware k DVB-S karte TT-connect S-2400 a schému zapojenia elektrických súčiastok.